NASA-CR-194566

RICIS Seminar Series

NCC9-16

PROCEEDINGS

Information Security and Integrity Systems

(NASA-CR-194566) INFORMATION SECURITY AND INTEGRITY SYSTEMS (Houston Univ.) 267 p

N94-70074

Unclas

Z9/61 0193090

May 15 - 16, 1990

University of Houston-Clear Lake
Bayou Building

Co-Sponsored by

NASA/Johnson Space Center
Computer Sciences Corporation
University of Houston-Clear Lake

SEPEC Seminar Series

PROCEEDINGS

Information Security and Integrity Systems

May 15 - 16, 1990

University of Houston-Clear Lake
Bayou Building

Co-Sponsored by

NASA/Johnson Space Center
Computer Sciences Corporation
University of Houston-Clear Lake

INTRODUCTION

Welcome to ISIS 1990

With the evolution of technology, computer literacy and accessibility have become commonplace in our society. The mass marketing and low price of personal computers, the simplification of programming and the availability of pre-packaged software have been instrumental in integrating the computer into everyday life. Nowhere is the integration of the computer and technology more evident than space programs. However, for those of us involved in the use and application of computers and technology, there are serious consequences if the resulting automated systems do not have high levels of integrity and availability. As computers perform more and more critical services, the most serious security concerns often become a matter of an assurance that the computer performs its critical functions correctly and that there are no harmful side effects.

A reasonable level of security sufficiency requires the implementation of management control processes and the integration of security in the development and use of the technology. Therefore, security should be an integral part of the entire planning, development and operation of automated systems. Much of what needs to be done to improve security is not clearly separable from what is needed to improve the usefulness, reliability, effectiveness, and efficiency of automated sytems.

ISIS 1990 provides a forum for distinguished professionals from industry, government, and universities to present attendees with the broadest possible exposure to the comprehensive field we know and information security.

We enjoin all participants, speakers and attendees, to openly discuss their views and experiences and continue the "networking" process that begins with this important symposium.

Rodney L. Bown
Technical Co-Chair

Frederick G. Tompkins
Technical Co-Chair

Table of Contents

Information Security and Integrity Systems

Information Security and Integrity Systems

Tuesday, May 15

8:30 a.m.

Welcome and Arrangements

Technical Co-Chairs: **Rod Bown**, *University of Houston-Clear Lake*
F. G. Tompkins, *Computer Sciences Corporation*
NASA/Johnson Space Center

9:00 a.m.

Tutorial: Common Session — Auditorium

A Computer Security Overview

Lance Hoffman

10:15 a.n
Break

Dr. Lance Hoffman is a professor of Computer Science at The George Washington University in Washinton, DC. He is the author or editor of three books and numerous articles on computer security with a fourth on computer viruses to be published this summer.

Noon

Lunch Speaker — Atrium II

Computers and the Law

Michael Gemignani
University of Houston-Clear Lake

Dr. Michael Gemignani is the Senior Vice President and Provost at the University of Houston-Clear Lake. He has written and spoken widely in the area of computer related law.

Time	Managerial Room 2-532	Technical Room 2-515
1:30 p.m.	**Evolution - User Computing Security** **Emily Lonsford** *MITRE*	**Trust: Formal Methods and Associated Techniques** **Susan Gerhart** *Microelectronics Computer Corporation (MCC)*
2:15 p.m.	**Security in Software Applications and Development** **James Molini** *Computer Sciences Corporation*	**Secure Distributed Operating System and Verification** **Doug Weber** *Odyssey Research Associates*
3:00 p.m. Break 3:30 p.m.	**Risk Management** **F. G. Tompkins** *Computer Sciences Corporation*	**Trusted Ada** **John McHugh** *Computational Logic Inc.*

5:00 p.m.

Wine & Cheese Reception — Atrium II

Wednesday, May 16

Time	Managerial Room 2-532	Technical Room 2-515
9:00 a.m.	**Contingency Planning** **Elmer Bomlitz** *Harris Devlin Associates*	**A Conceptual Model for Supporting B3+ Dynamic Multilevel Security and Integrity in the Ada Runtime Environment** **Charles W. McKay** *University of Houston-Clear Lake*
9:45 a.m. Break 10:15 a.m.	**Information Security Program Development** **James R. Wade** *Battelle Memorial Institute*	**Complexity Issues** **Howard Johnson** *Information Intelligence Sciences*
11:00 a.m.	**Investigating Computer-Based White-Collar Crime** **Neal Findley** *U. S. Secret Service*	**Security in Computer Networks** **Colin Rous** *Digital Equipment Corporation*

Noon

Lunch Speaker — Atrium II

Ethics: Mandate VS. Choice

Marlene Campbell

Dr. Marlene Campbell is an assistant professor of Computer Science at the Murray State University in Murray, Kentucky.

1:30 p.m. — Auditorim

Computer Viruses
Angel Riveria
Sector Technologies, Inc.

3:00 p.m.
Break
3:30 p.m.

Closing Panel — Auditorim

Panel discussion on NASA concerns related to trusted computing support for life and property critical systems

Information Security and Integrity Systems

General Information

Time	May 15, 1990 8:45 a.m.- 6:00 p.m. May 16, 1990 9:00 a.m. - 4:30 p.m.
Location	University of Houston-Clear Lake Bayou Building
Cost	$250 Industry $150 Government/University $60 Student Price includes presentations, abstracts, refreshments, 2 lunches, and materials. Seating is limited to 250 people.

SEPEC

Software Engineering Professional Education Center
University of Houston-Clear Lake
2700 Bay Area Boulevard, Box 258
Houston, Texas 77058
(713) 282-2223 phone
(713) 282-2249 fax

Tutorial

A Computer Security Overview

Lance Hoffman

A COMPUTER SECURITY OVERVIEW

Prof. Lance J. Hoffman
The George Washington University
Dept. of Electrical Engineering and Computer Science
(202) 994-4955
hoffman@gwusun.gwu.edu

GOALS OF THIS TUTORIAL

- Review security requirements imposed by government and by common sense
- Examine risk analysis methods to help you keep sight of forest while in trees
- Discuss the current hot topic of viruses (which will stay hot)
- Examine network security, now and in the next year to 30 years
- Give a brief overview of encryption
- Review protection methods in operating systems
- Review database security problems
- Review the Trusted Computer System Evaluation Criteria (Orange Book)
- Comment on formal verification methods
- Consider new approaches (like intrusion detection and biometrics)
- Review the old, low tech, and still good solutions
- Give pointers to the literature and to where to get help

COMPUTER SECURITY ACT OF 1987

(courtesy of Social Security Admin.)

- Purpose: improve security and privacy of sensitive info in government systems
- Purpose: create means for establishing minimum acceptable security standards
- Tasks NBS (NIST) with developing standards and guidelines for S&P
- Provides for promulgation of such standards and guidelines
- Operators of federal computer systems with sensitive info need security plans
- Mandatory periodic training for all who manage, use or operate such systems

- Main NIST purpose: control loss and unauth. modification or disclosure
- ... and to prevent computer–related fraud and misuse
- Also establishes a S&P advisory board within Commerce Dept. to advise

SENSITIVE INFORMATION

"any information the loss, misuse or unauthorized access to or modification of which could adversely affect the national interest or the conduct of Federal programs or the privacy to which individuals are entitled by the Privacy Act"

Computer Security Act of 1987

COMPUTER SECURITY ACT OF 1987

Timetable for Agencies

- Within 6 months of enactment, identify each system with sensitive information.
- Within a year, establish a plan for S&P of such systems.
- Send plans to NBS (NIST) and NSA for advice and comment.
- Include a summary of the plan in the Agency's 5-year plan approved by OMB.

OMB Circular A-130

Management of Federal Information Resources

Excerpts from Policies:

1) Collect only necessary information
2) Don't invade personal privacy or violate confidentialities
3) Provide individuals with access to information and amending principles as laid out in the Privacy Act
4) Establish security for information systems commensurate with risk and magnitude of loss or harm resulting from improper operation

Excerpts from Appendix I to A-130:

1) Privacy Act Annual Reports: any Privacy Act inquiry should generate a log of how the request was handled (see appendix for details)
2) A Federal Register publication is required if a system is new or altered in a significant way, e.g., a change in number or types of individuals on whom records are maintained; an expansion of types of information maintained; a change in the purpose for which the information is used; a change that creates substantially greater access to records in the system (e.g., putting remote terminals in field offices for a formerly HQ-only system). Details in appendix.

WHAT IS RISK ANALYSIS?

- An emerging analytic discipline, consisting of two parts:
- -- *Risk assessment:* determining what the risks are
- -- *Risk management:* evaluating alternatives for mitigating the risk

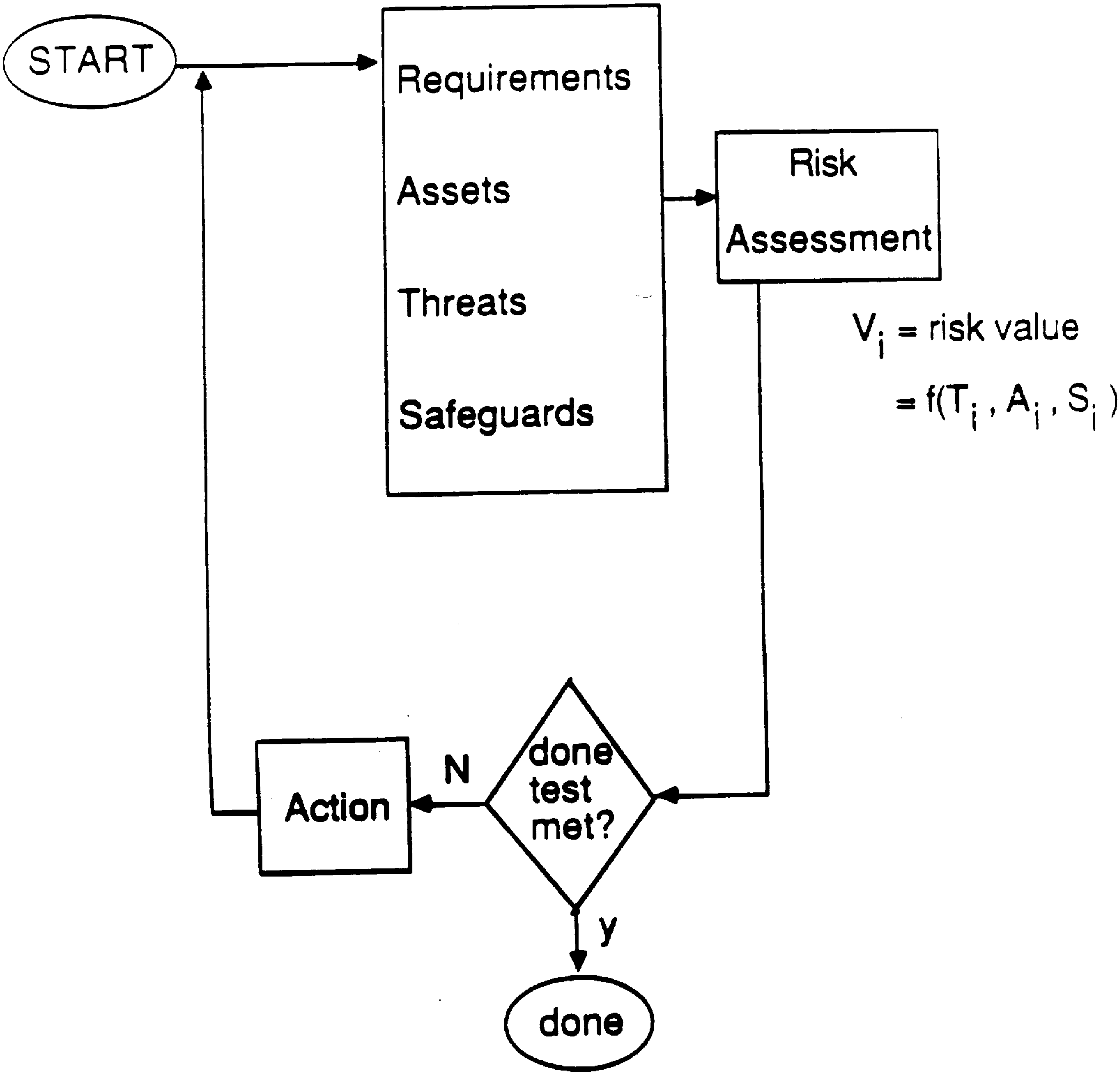
RISK ANALYSIS

Risk Assessment

- Determine Risks
- Estimate exposure of (computer) resources to loss
- Consider assets, threats, vulnerabilities
- Typically computed using asset values, threat likelihoods, CM effectivenesses
- Can be Simple Self-Analysis or Complex and Done by Outsiders
- Considers Potential Losses (Both Dollars and Goodwill)
- Should Indicate Where to Most Effectively Use Your Limited Resources

- Countermeasures
- Countermeasure selection
- Sensitivity analysis
- Decision analysis
- Goal-seeking heuristics
- Risk perception and communication

RISK MANAGEMENT FRAMEWORK



ASSETS

- People and skill
- Goodwill
- Hardware
- Software
- Data
- Documentation
- Supplies

THREATS AND VULNERABILITIES

- Disclosure
- Destruction
- Modification
- Denial of service

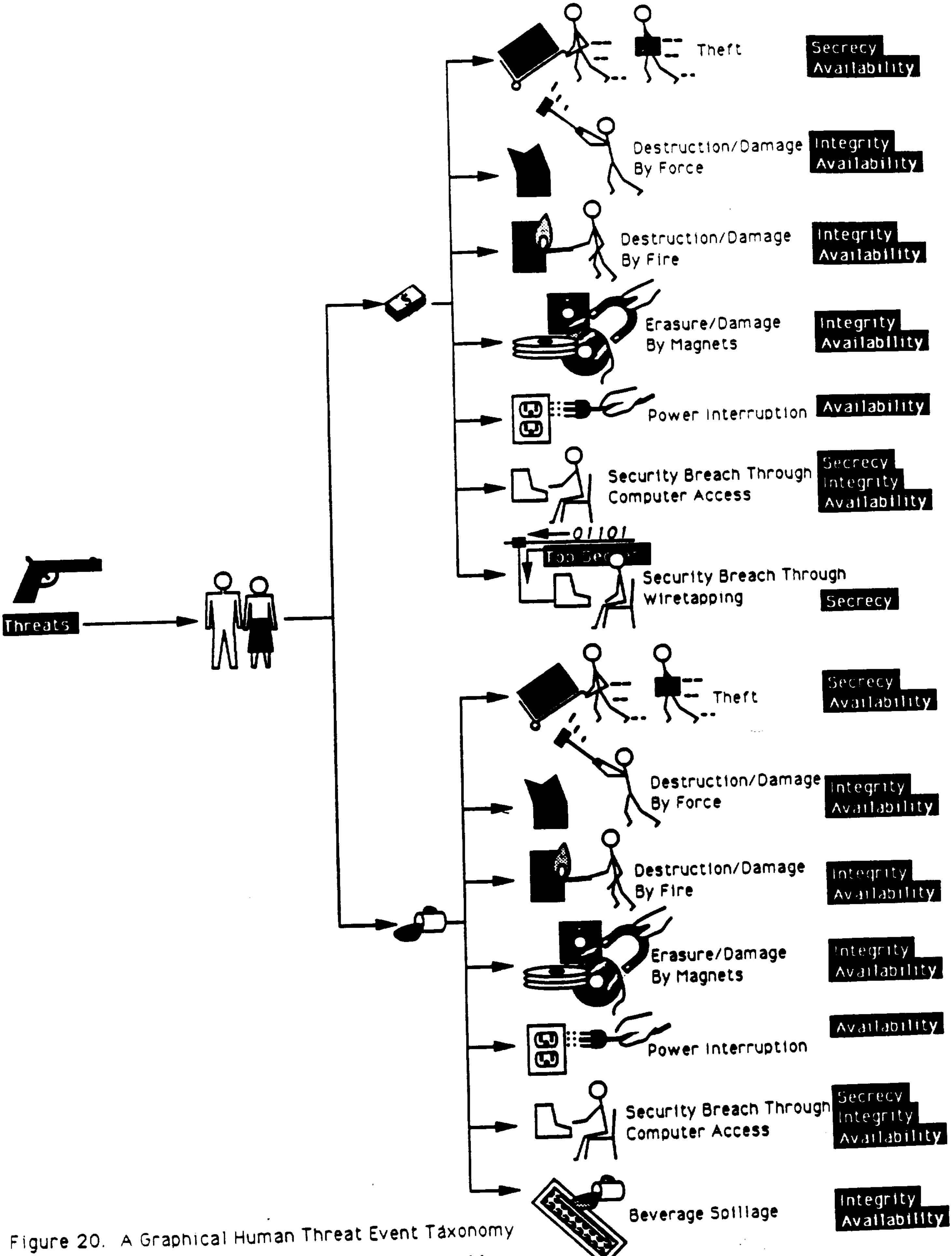


Figure 20. A Graphical Human Threat Event Taxonomy

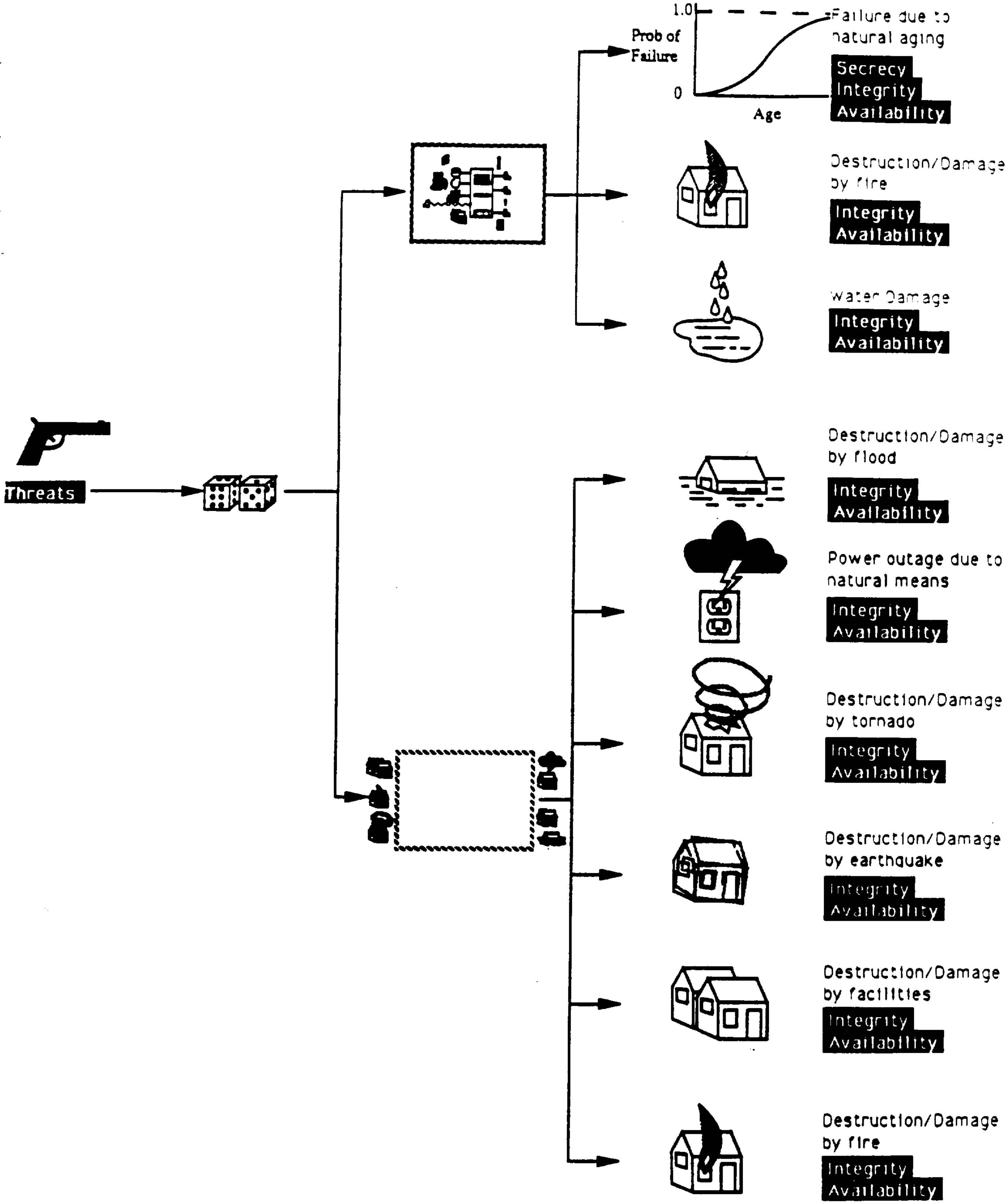


Figure 21. A Graphical Natural Threat Taxonomy

STARTING AND STOPPING

- DON'T ...
- avoid due to fear of cost; you may be losing more (in many ways) by delaying
- slavishly do a full fledged FIPS 65 analysis if not called for

- DO ...
- secure management commitment for a certain level of resources
- act breadth–first rather than depth–first

RISK ASSESSMENT METHODOLOGIES

- Expected values
- – matrices and fault trees
- Worst case
- Checklists and questionnaires
- Fuzzy (qualitative) risk analysis
- Hybrid methods
- Use and applicability of automated packages
- Current efforts to develop a general model

TYPICAL R.A. METHODOLOGIES

- – FIPS 65 Expected Values
- – Simplification to 7–point scale
- – Variants to produce R.O.I., etc.
- – Fault trees (quantitative or qualitative)
- – Kepner–Trego weights (quantitative or qualitative)

NIST (NBS) FIPS 65 METHODOLOGY

- Define system ASSETS (data files, eqpt., negotiable output, etc.)
- Define THREATS (leading to unauth. destruction, disclosure, mods, denial)
- FOR EACH ASSET
- FOR EACH THREAT
- Estimate frequency of THREAT to ASSET
- Estimate dollar loss if realized
- Multiply freq*loss to obtain ANNUAL LOSS EXPECTANCY for threat/asset pair
- (SUM OVER ALL ASSET/THREAT PAIRS TO OBTAIN SYSTEM–WIDE *A.L.E.*)

QUANTIFYING THE RISKS

ANNUAL RISK EXPOSURES AND THREAT EFFECTS

(Based on Dept. of Agriculture methodology)

THREAT	LOSS CATEGORY: DESTRUCTION	DELAY	DISCLOSURE	MODIFICATION	OTHER	TOTAL SINGLE-TIME LOSS	OCCURENCE RATE	ANNUAL RISK EXPOSURE
NATURAL HAZARDS WIND-STORMS		$9.025				$ 9.025	.4	$ 3.[illegible]
SEVERE STORMS		$9.025				$ 9.025	1	$ 9.025
EARTH-QUAKES	$435.725	$52.225				$487.950	.03	$14.639
FLOODS							Impro-bable	...
ACCIDENTS MINOR FIRE-COMPUTER ROOM	$ 157.770	$5.700			$2.500	$165.970	2	$33.[illegible]94
MAJOR FIRE-COMPUTER ROOM	$1.577.700	$52.225			$29.938	$1.659.863	.07	$116.[illegible]90

NAVY RAM ANNUAL LOSS EXPECTANCY COMPUTATION

ANNUAL LOSS EXPECTANCY COMPUTATION

ASSET IMPACT VALUE RATING

SUCCESSFUL ATTTACK FREQUENCY RATING		< $10 (1)	$100 (2)	$1000 (3)	10^4 (4)	10^5 (5)	10^6 (6)	10^7 (7)	10^8 (8)
ONCE EA. 300 YRS	1	0	0	0	0	$300	3K	30K	300K
30 YRS	2	0	0	0	$300	3K	30K	300K	3M
3 YRS	3	0	0	$300	3K	30K	300K	3M	30M
100 DAYS	4	0	$300	3K	30K	300K	3M	30M	300M
10 DAYS	5	$300	3K	30K	300K	3M	$\boxed{30M}$	300M	•
ONCE/DAY	6	3K	30K	300K	3M	30M	300M	•	•
10 TIMES/DAY	7	30K	300K	3M	30M	300M	•	•	•
100 TIMES/DAY	8	300K	3M	30M	300M	•	•	•	•

courtesy of DODCI

$$ALE = \frac{10^{(I + F - 3)}}{3}$$

CHECKLISTS AND QUESTIONNAIRES

So Why Doesn't Everybody Use Them?

- No real measure of total exposure or exposure by area
- No guidance on what gaps to plug first or to ignore

BASIC STEPS OF RISK ANALYSIS

(Pfleeger 1988)

- Identify assets
- Determine vulnerabilities
- Estimate likelihood of exploitation
- Compute expected annual loss
- Survey applicable controls and their costs
- Project annual savings of control

TYPICAL SAFEGUARD CATEGORIES

(Pfleeger 1988)

- cryptographic controls
- secure protocols
- program development controls
- program execution environment controls
- operating system protection features
- identification
- authentication
- secure operating system design and implementation
- Data base access controls
- Data base reliability controls
- Data base inference controls
- Multilevel security controls for data, data bases, and operating systems
- Personal computer controls
- Network access controls
- Network integrity controls
- Controls on telecommunications media
- Physical controls

RISK ANALYSIS

Risk Management: Safeguard Selection

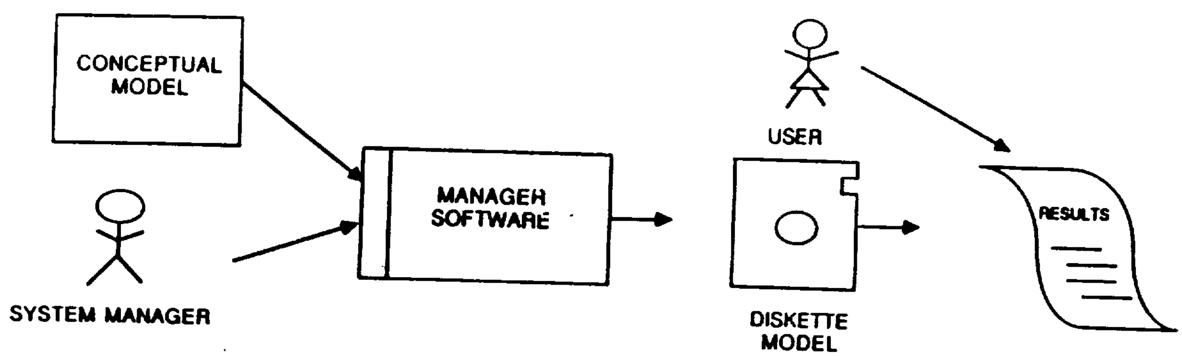
- Select safeguards and maximize exposure reduction, given real world constraints:
- -- political
- -- technical
- -- monetary
- Can now use automatic what-if? (computer is much faster than human)
- Good for real world NON-LINEAR models of real situations

AUTOMATED RISK ANALYSIS

- Now usually PC-based
- Typically *not* spreadsheets only, since they're unfriendly & require model setup
- A few methodologies and packages are used
- Reduces level of effort by providing standard report and doing arithmetic
- Allows first level assessments to be done cheaply in-house
- Useful in increasing security awareness
- Safeguard selection is not as far along, and often requires a trained analyst

CONTROL OF LEVEL OF EFFORT

- One person or a small team may be able to build an organization-wide model
- This may be based upon canned models supplied and training/consultation
- After model is set up with appropriate reports and fixed values and formulas,
- It can be sent to lower levels to fill in without fear of alteration



29

SYSTEM MANAGER CREATES MODEL, USER INVOKES IT.

NO MAGIC BULLET

Expectations from Automated Packages

- IN MOST CASES, you can't put the diskette into the computer and be done.
- IN MANY CASES, the $50–$500 "user version" won't be adequate for you.
- This is true even though some systems come with various "models" to use.
- OFTEN the only alternative to a shoddy or no job is to work with the vendor ...
- TAILORING the package to your specific situation. Expect to pay for this.
- MORE LIKELY SCENARIOS: Buy/rent "system manager, vendor trains you.
- Cost depends on how knowledgable you are to start and the size of your job.

DON'T BUY JUST ANY PACKAGE!

- Creativity, judgment, and accuracy of risk model *not* guaranteed!
- Package should supply computational support and flexibility for *your* needs.
- As always, beware of computer–aided reports appearing very impressive.
- Guidance: Ellis and Garrabrants thesis from the Naval Postgraduate School
- Guidance: NIST Risk Analysis Lab (one stop hands on shopping) (Irene Gilbert)

BUILD YOUR OWN VS. EXPERT HELP

(Make vs. buy?)

BUILD YOUR OWN MODEL	HAVE EXPERT BUILD IT
No extra $ required	Costs: $, time to locate
Much more of your time	Much less of your time
Must test before using	Can be more off-shelf
Knowledge must be here	Use other('s) knowledge
Long time to completion	Can demand at fixed time

RISK ANALYSIS SUMMARY REPORT FORMAT

(adapted from USDA)

Table of Contents

Exhibits

1. **Table: Existing Safeguards Related to Threats**

2. **Table: Safeguards Being Implemented Related to Threats**

3. **Discussion of Recommended Safeguards**

IV. Risk Analysis

A. Published guidelines used

B. Major threats considered and why

C. Worksheets and summary

D. Countermeasures (with costs); cost-benefit analysis of each countermeasure/ threat combination

I. Introduction

A. Reason for risk analysis study and its scope

B. Description of physical facility

C. Major security measures in use or being installed

III. Requirements and Constraints

A. Historical factors (previous risk analyses and results, serious security breaches, etc.)

B. Time and manpower considerations; other constraints

Model Risk Analysis Report

- DRAFT -

Most of the projected security losses result from [insert appropriate number] security vulnerabilities. The most important vulnerability is [name the most significant vulnerability]. [Name the second and third vulnerabilities] are also important. [Now describe the strong points of the Facility's security program briefly.]

The risk analysis indicates that Threat1 is the most serious threat to the FacilityX. The Threat1 Annualized Loss Expectancy (ALE) is estimated to be $nnn,nnn per year, which is XX% of the total ALE. This loss exposure is largely due to [describe briefly the major vulnerabilities that account for the loss]. [Name Threat2 and Threat3] also represent serious loss exposures of $aaa,aaa per year and $bbb,bbb per year respectively. [Include the following, or a similar sentence if appropriate.] While the ALE of Threat4 is relatively low, its single occurrence, loss (the estimate of the loss that would result from a single occurrence of the threat) is $nnn,nnn per occurrence which would have a material impact on the budget of the FacilityX.

The analysis has led to (not more than four or five) major recommendations:

1) recommendation one,
2) recommendation two,
3) recommendation three, etc.

The cost to implement all the recommendations presented in Section 3 is $nnn,nnn. It is estimated that these recommended security measures will reduce the total ALE of the FacilityX from $nnn,nnn per year to $mmm,mmm, a return on investment of about zz%.

[Insert the sentence at the beginning of the next paragraph into the beginning of whichever paragraph you select as the first "assets" ALE paragraph. Note also that the paragraph is worded as though it were the paragraph selected to appear first. The remaining paragraphs are worded more tersely.]

(U. S. State Dept.)

RISK ANALYSIS WORK PLAN (NIH)

1. Develop project plan, brief participants.
2. Identify replacement costs for tangible assets.
3. Develop single-time loss estimates for related procedures.
4. Analyze threats, develop threat occurrence rates.
5. Analyze vulnerabilities, document existing safeguards.
6. Calculate annual loss exposures.
7. Identify potential safeguards.
8. Cost-benefit analysis of potential safeguards.
9. Develop recommended safeguards.
10. Produce final report.

PITFALLS TO AVOID

- **Appearing Indifferent to Human or Political Costs: "Too Analytical"**
- **Addressing Critical Issues with Fuzzy Data AND Crisply Computed Answers**
- **User Misinterpretation of Results -- Education Needed**
- **Scope Selection – Must be Tailored to Schedule and Team Size**
- **Misconfirmed Findings**
- **Team Qualifications**
- **Failure to Get Management Involved and Visible**
- **Rush to Design and Procure Safeguards**
- **Overemphasizing Sophisticated, Expensive Solutions**
- Using Checklists with Hit-and-Miss Safeguard Selection

TYPES OF ROGUE PROGRAMS

- Virus
  - a program that attaches itself to other programs and reproduces itself in the process
- Worm
  - a program that reproduces itself and propagates into other systems without attachment to or infection of another program
- Trojan Horse
  - a program that performs some unexpected hidden function
- Logic Bomb
  - a piece of code hidden within another program that check for some logical condiction before executing some unexpected condition (example: IF Fred is no longer in employee-data-base THEN erase all files)
- Time Bomb
  - a logic bomb triggered by a condition based on time (e.g., IF today = "Dec 25" THEN draw Christmas-tree)

IPM PC VIRUS GROWTH

A. 1986

- 1 new virus: Brain

B. 1987

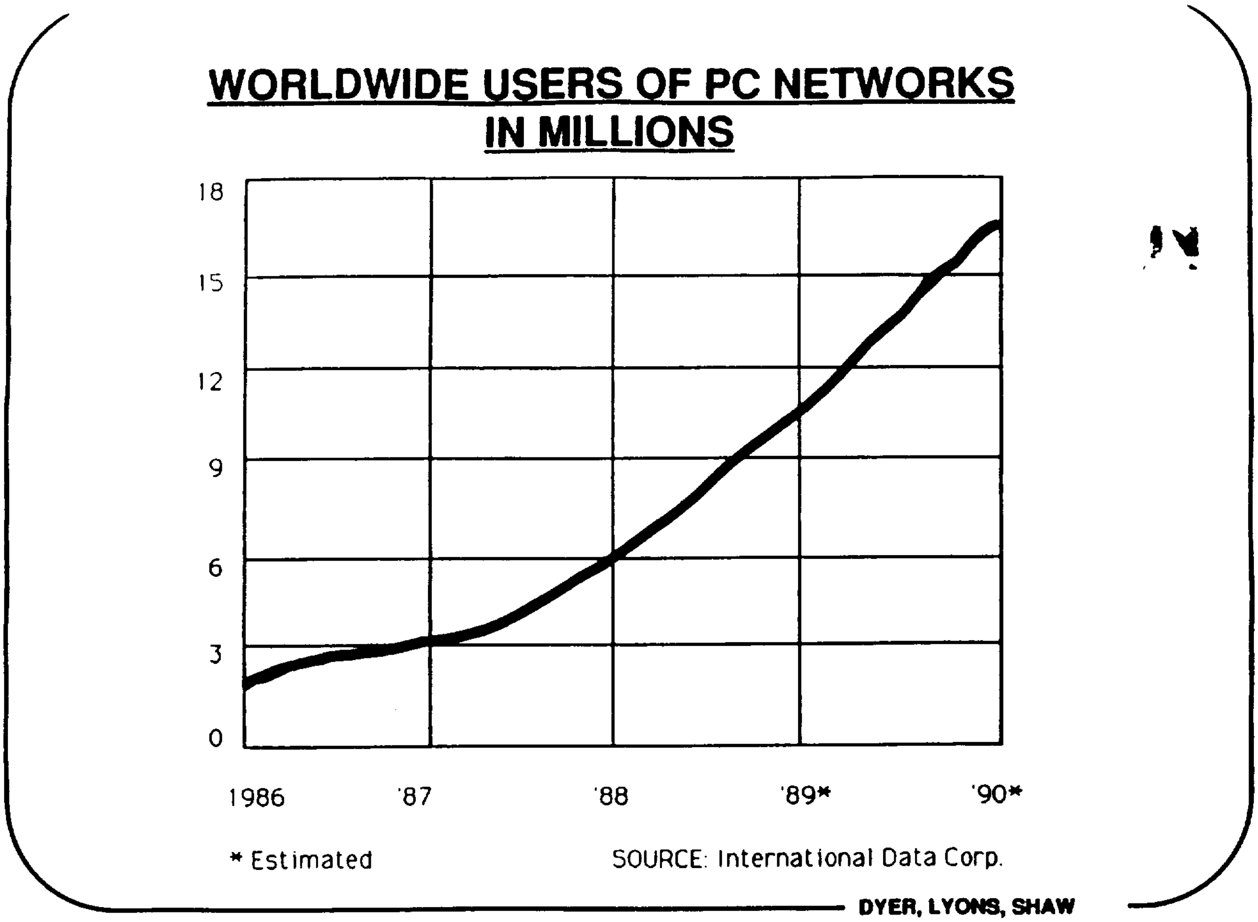
- 5 new viruses: Alameda, S. African, Lehigh, Vienna, Israeli

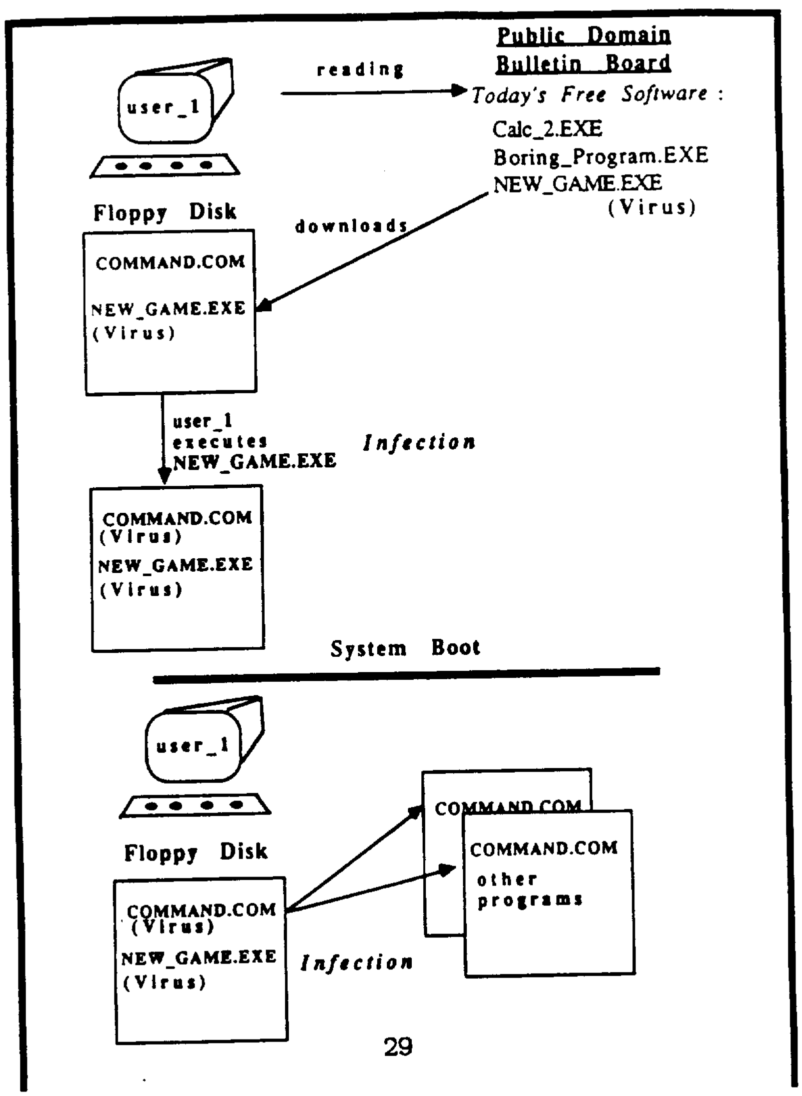
C. 1988

- 5 more: Italian, Dos 62, New Zealand, Cascade, Agiplan

D. 1989

- 10 at least: Oropax, Search, dBase, Screen, Datacrime, 405, Pentagon, Traceback, Icelandic, Mistake


WORLDWIDE USERS OF PC NETWORKS IN MILLIONS
18
15
12
9
6
3
0
1986
'87
'88
'89*
'90*
* Estimated
SOURCE: International Data Corp.
DYER, LYONS, SHAW


Public Domain Bulletin Board
Today's Free Software :
Calc_2.EXE
Boring_Program.EXE
NEW_GAME.EXE (Virus)
user_1
reading
downloads
Floppy Disk
COMMAND.COM
NEW_GAME.EXE (Virus)
user_1 executes NEW_GAME.EXE
Infection
COMMAND.COM (Virus)
NEW_GAME.EXE (Virus)
System Boot
user_1
Floppy Disk
COMMAND.COM (Virus)
NEW_GAME.EXE (Virus)
Infection
COMMAND.COM
COMMAND.COM
other programs

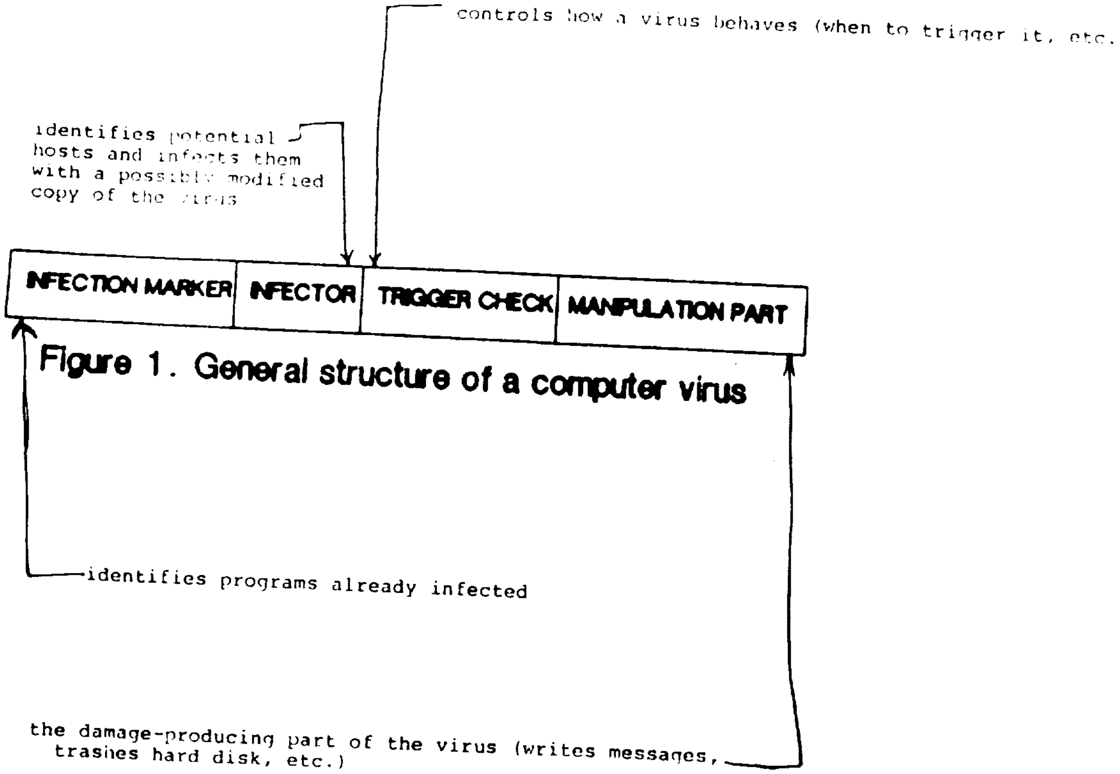


Figure 1. General structure of a computer virus

from Burger, Computer Viruses: A High-Tech Disease

TRIGGER DATES FOR SOME VIRUSES

Date	Virus	Effect
12th October onward any year	Datacrime	Message and disk format
Friday the 13th any year	South African	File deletion
	Israeli	File deletion
April 1st	April-1-COM	Lock up system
	April-1-EXE	Lock up system
March 2nd 1988	Peace	Message and self-deletion
October-December 1988	Cascade	Cascade display
December 5th 1988 onwards	Traceback	Direct file infection
December 28th 1988 onwards	Traceback	Cascade display
August 1989 onwards	FuManchu	Character substitution
Friday the 13th 1990 or later	Jerusalem-D	Destroys FATs
Friday the 13th 1992 or later	Jerusalem-E	Destroys FATs
1st January 2000	Century	Destroys FATs and sectors

(Spafford)

PC ARCHITECTURAL VULNERABILITIES TO VIRAL ATTACKS

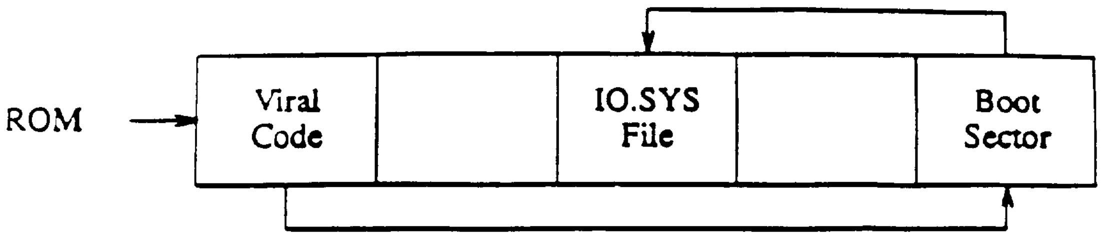
- Operating System loaded from disk boot sectors
- User capable of modifying system interrupt vector and working storage managment fields
- All disk sectors (including FATs) modifiable by users
- Physical write-protection only available to the disk level, not at the track or sector level

Boot Infectors

- Attach to boot sectors of floppy and/or hard disks
- Gain control of system when it is powered on
- Typically stay memory resident
- Must have at least their initial portions in specific locations on disk
- Can infect any disk subsequently inserted in machine (since memory res.)

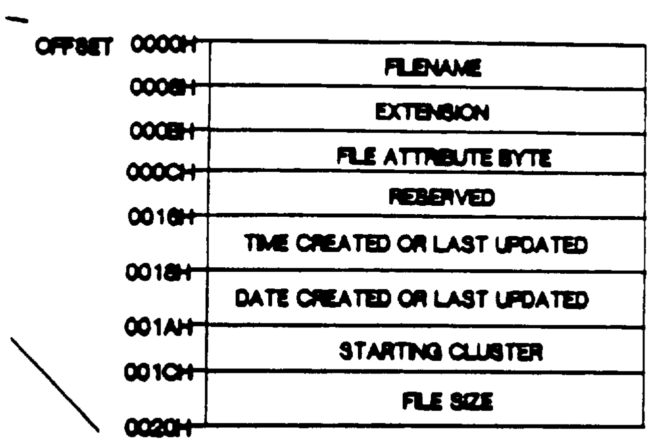
THE IBM PC BOOT SEQUENCE

- ROM BIOS routines
- Partition record code execution
- Boot sector code execution
- IO.SYS and MSDOS.SYS code execution
- COMMAND.COM shell execution
- AUTOEXEC.BAT batch file execution



After Alameda Virus Infection

(Spafford)



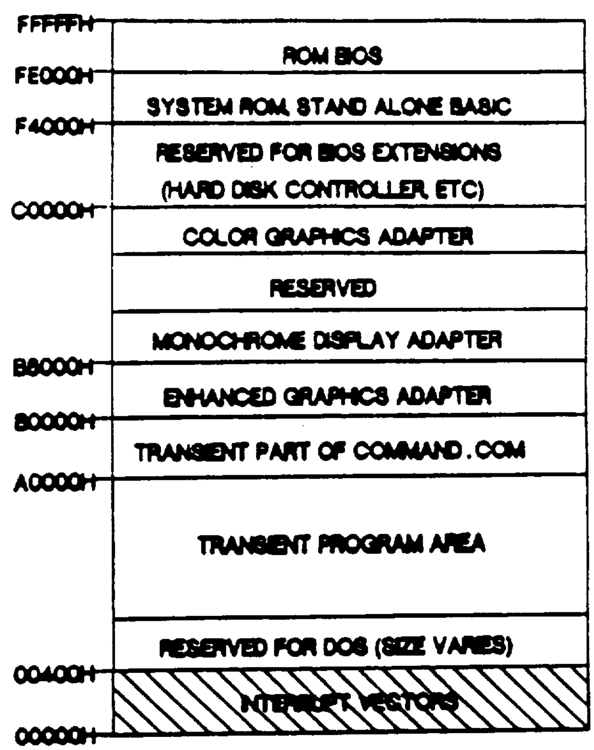
Layout of root directory entry

(from Duncan [5])

These can be modified, like any other data on disk.
So a file can be infected and appear untouched ...
... from the outside.

SYSTEM INFECTORS

- Attach to at least one operating system module or device driver (e.g. COMMAND.COM)
- Gain control during system initialization following boot sequence
- Only able to infect specific files
- But these files present on many machines and thus provide standard attack point
- All well-behaved programs process their requests for system services (like disk reads and writes) through infected attack point (e.g., COMMAND.COM)



System memory map of IBM-PC
(from Duncan [5])

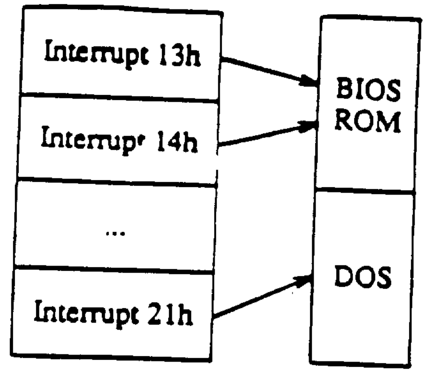
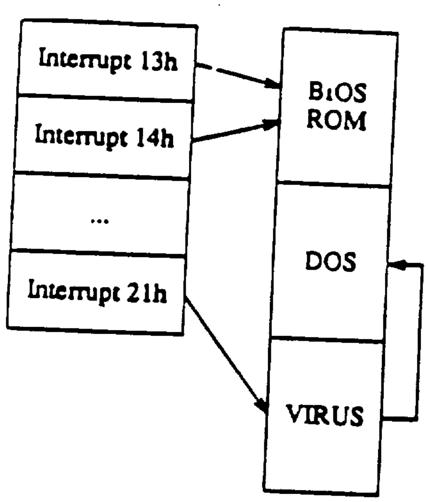


Figure 3.6: Normal interrupt usage

Interrupts commonly used by viruses (values in hexadecimal)

8	System timer (called 18.2 times a second)
9	Keyboard interrupt
13	BIOS floppy disk input/output
17	Printer interrupt
19	System warm boot
1C	System timer (secondary interrupt)
21	DOS service call
25	Absolute disk read interrupt
26	Absolute disk write interrupt
27	Terminate and stay resident
28	Keyboard busy loop
70	Real time clock interrupt

(Spafford)



CASE HISTORIES OF VIRUSES

- BRAIN: Set up to propagate internationally
- SCORES: Attacked internal programs at a major corporation
- SHRINK WRAP: Commercial software accidentally propagated a virus
- ISRAELI: Some considered a political weapon
- INTERNET WORM: Massive denial of service around country through a trap door

COMPUTER VIRUS CASE HISTORY

Brain

THE PAKISTANI VIRUS

- DISTRIBUTION:

- **THE VIRUS WAS DISTRIBUTED ON BOOTLEG VERSIONS OF MS-DOS SOFTWARE SOLD IN PAKISTAN.**

- CREATION:

- **IT WAS CREATED BY 19-YEAR OLD BASIT ALVI.**

- EFFECTS:

- **IT REPLACED BIOS DISK INTERRUPT, AND WHENEVER THIS BIOS CALL WAS MADE IT INFECTED ALL DISKS ON THE SYSTEM.**
- **IT WROTE 'WELCOME TO THE DUNGEON' ON THE BOOT SECTOR OF A DISK, RENDERING IT UNREADABLE.**
- **IT WROTE 'BRAIN' ON DISK LABEL**
- **IT INFECTED 100,000 DISKS IN US, AND AS MANY AS 10,000 AT GEORGE WASHINGTON UNIVERSITY.**

EXAMPLE OF "SCORES" MAC VIRUS

- Non-overwriting, infects applications and system file resources
- Dormant first two days
- System file resources loaded and run at boot time, so virus becomes ...
- a memory-resident part of operating system!
- Applications using VULT and ERIC resources are attacked every 3.5 minutes
- (Virus doesn't like proprietary software written by Electronic Data Systems)
- Lets these applications run 25 minutes, then bombs them
- 7 days after infection, causes disk writes from VULT to fail after 15 min.

(Mad Macs, MacWorld 11/88, RP)

INTERNET "WORM"

- Exploited BSD 4.2 Unix and utility program flaws (features?)
- Bug caused replication much faster than intended, jamming 2,000–6,000 computers
- Used a dictionary attack against 1-way encrypted passwords stored unprotected
- Used an integrity bug (buffer overrun)
- Robert Morris, Jr. convicted Jan. 1990 under 1986 Cptr Fraud & Abuse Act

THE INTERNET WORM

- C program released in November 1988, propagated to many Internet hosts
- So busy propagating, it tied up the net
- Could have done major damage: delete or modify existing files, record passwords
- Did not invoke superuser privileges

THE INTERNET WORM

Overview

A. Two parts
- Main program
- Bootstrap (Grappling Hook)

B. Main Program
- Exploited host for data related to remote hosts
- Initiated atacks
- Acted as server to infected hosts

C. Bootstrap
- Infected remote hosts
- Compiled, linked, ran on remote hosts
- Retrieved main program onto host
- 99 lines of C code

D. Camouflaged

Bootstrap Attacks via Trap Doors

A. Electronic Mail
- Exploited SENDMAIL's DEBUG option

B. DEBUG Option
- Allows sequence of commands to be sent as mail message
- This sequence instructed host to strip header, pass body to..
- ... command interpreter, which caused the worm source in body
- ... to be compiled, linked, and executed.

C. Most machines compile SENDMAIL with DEBUG *ON* (many still do)

D. Most binary versions also delivered with DEBUG enabled

(courtesy Voreh,Crider,Reagan)

INTERNET WORM

Bootstrap Attacks via Trap Doors

A. System Query BUG in 'finger' command
   - FINGER did no range checks on command parameters
   - Worm passed large buffer, overflowing command buffer

B. REMOTE SHELL allows access to trusted host w/o password check

C. REMOTE EXEC allows remote execution given name and passwd
   - Morris attempted REXEC connections with guessed passwd
   - Guesses: null, no p/w, username, name backwards and appended
   - Guesses: 432 word list of common passwords

Internet Virus Lessons Learned

(Eichin and Rochlis CACM 6/89)

- Connectivity important: can't get timely fixes if off the network
- Old boy (trusted) network worked
- Late night authentication (of fixes) a problem
- Whom do you call to find mgr of OSU comp center from MIT at 3 a.m.?
- Speaker phones and conference calling very helpful
- Misinformation and illusions run rampant
- Tools were not that important (hand decompiled)
- Source availability was important
- Academic sites performed best
- Hard to work with press hounding you. MIT ++, Berkeley --

Internet Worm Open Issues

(Eichin and Rochlis CACM 6/89)

- Ignoring Least Privilege Principle Left This Door Open
- Author was an insider, so many procedural and tech CMs would fail
- Backups good. Don't make cure worse than disease. $ to protect too high?
- Defenses MUST be at host, not at network
- Logging info is important
- Denial of service attacks are easy
- A central security fix source may be a good idea
- Avoid knee jerk reactions

CERT Contact Information

For Emergencies:

(412) 268-7090

For Information:

(412) 268-7080

Electronic Mail:

cert@sei.cmu.edu

U.S. Mail:

CERT/CC
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

SAFE COMPUTING PRACTICES

(Be careful but be reasonable.)

- Mandatory: BACKUP, BACKUP, BACKUP!
- Discretionary: boot from hard drive, minimize BBS use, don't loan disks,...
- ... don't execute programs of unknown origin, install new s/w on isolated sys.
- ... Make command and executable files read only, remove from root directory...

PROTECTION METHODS

- Safe Computing Practices
- User Awareness and Policy Guidelines
- Software Protection
- Hardware Protection

SOFTWARE PROTECTION

- Monitor system activities (TSR programs typically)
  - executable files
  - configuration files
  - operating system functions
  - device drivers
  - boot blocks
  - RAM

- Perform other useful functions
  - initiate password protection
  - screen program execution
  - perform file management functions
  - maintain operational logs

- Limitations
  - produce false alarms
  - cannot detect all viruses

DYER, LYONS, SHAW

SOFTWARE PROTECTION PRODUCTS

	C-4	CERTUS	DISK WATCHER	DR PANDA UTILITIES	FLU SHOT +	MACE VACCINE	PROTEC	VIRUS GUARD
MONITORS DOS INTERRUPTS	X	X	X	X	X	X	X	X
MONITORS READS	X	X		X	X		X	
MONITORS WRITES	X	X	X	X	X	X	X	
TOGGLE ON/OFF	X				X	X		X
PREVENTS FORMATS	X			X	X	X		
PROTECTS CRITICAL AREAS	X	X	X	X	X	X	X	X
OPERATIONAL LOG		X					X	
PASSWORD PROTECTION		X					X	
PROGRAM SCREENING	X	X		X	X		X	X
OPERATING SYSTEM	DOS	DOS	DOS	DOS	DOS	DOS	DOS	DOS
COST	39.95	189.00	99.95	79.95	10.00	20.00	295.00	24.95
RAM REQUIRED	12 K	512K	47 K	3 K	256 K	256 K	70 K	128 K

DYER, LYONS, SHAW

SOFTWARE DETECTION PRODUCTS

	DATA PHYSICIAN	FLU SHOT +	SAM	SOFTSAFE	VIRUSAFE	VIR-X
USES CHECKSUMS	X	X	X	X	X	X
FLAGS PROG WITH CHANGE SIZE	X	X	X			X
DETECTS ON DEMAND	X		X	X	X	X
DETECTS BEFORE PROG EXECUTION		X				X
PROGRAM INTEGRITY CHECKS	X	X	X		X	X
MEMORY INTEGRITY CHECKS	X	X	X		X	
CHECKS FOR SPECIFIC VIRUS			X		X	
PROTECTION FEATURES	X	X	X	X	X	X
REMOVES VIRUSES	X		X	X	X	
OPERATING SYSTEM	DOS/UNIX	DOS	MAC	DOS	DOS	DOS
COST	99.00	10.00	99.95	99.00	150.00	59.95
RAM REQUIRED	256 K	256 K	18 K	40 K	7 K	128 K

DYER, LYONS, SHAW

SOFTWARE IDENTIFICATION PRODUCTS

PRODUCT	OPERATING SYSTEM	VIRUSES IDENTIFIED	COST
ANTIVIRUS	MAC	NVIR	PUBLIC DOMAIN
DISINFECTANT	MAC	ALL MAC VIRUSES	PUBLIC DOMAIN
FERRET	MAC	SCORES	PUBLIC DOMAIN
IBM DATACRIME	DOS	DATACRIME	$ 35.00
INTERFERON	MAC	NVIR,SCORES, "SNEAK VIRUSES"	SHAREWARE
KILL SCORES	MAC	SCORES	PUBLIC DOMAIN
NVIR ASSASSIN	MAC	NVIR	PUBLIC DOMAIN
VIREX	MAC	PEACE,NVIR,SCORES	$ 99.95
VIRUS DETECTIVE	MAC	PEACE,NVIR,SCORES,HPAT, INIT 29, ANTI	SHAREWARE
VIRUS RX	MAC	ALL MAC VIRUSES	PUBLIC DOMAIN
VI-SPY	DOS	22 MS DOS VIRUSES	$ 250.00

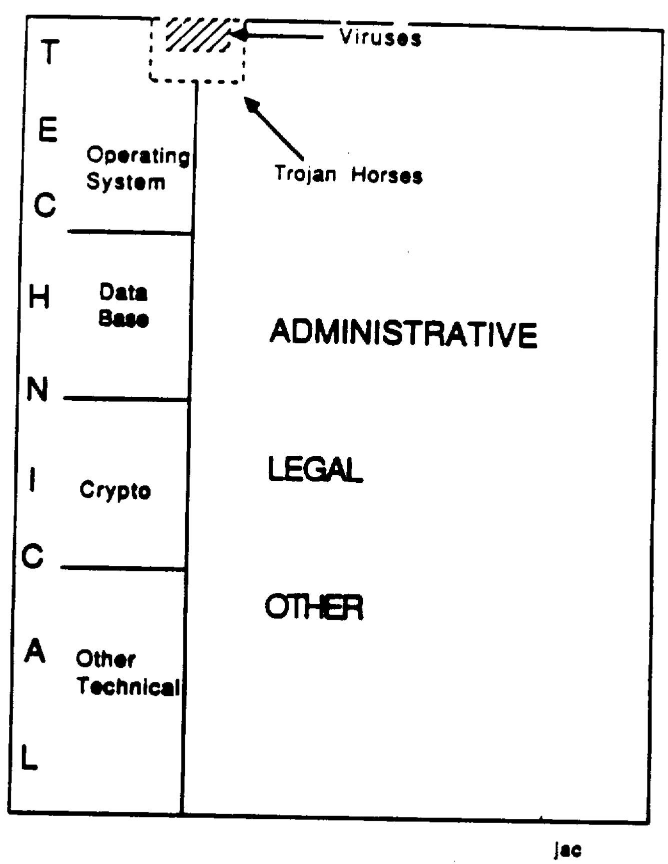
DYER, LYONS, SHAW

RECOVERY

- Notify those with whom you may have shared infected diskettes
- Use a disinfection utility
- Use backups

COMPUTER SECURITY AND VIRUSES:

Don't Miss the Big Picture!



THE FUTURE - TECHNICAL

- Automatic backups (exist now, more or less cumbersome)
- Automated configuration management by a separate processor
- Required unforgeable identification and authentication at terminals + better log
- -Maintain audit trail (liability chain)
- "Blessing" or "trusting" mechanisms before software runs on a system

LEGAL EFFORTS AGAINST RP's

- 1986 Computer Fraud and Abuse Act: crime to knowingly gain unauthorized access..
- ... to a govt. computer and cause abnormal operation (convicted Morris)
- 1988 legislation (HR55): crime to insert unauth. code or info that would cause..
- ... loss, through interstate commerce
- CA Sec 502 Penal Code: Individuals who author and/or knowingly distribute a ...
- ...virus face $10K fines, loss of eqpt, and three years in jail
- COMPENDIUM OF STATE AND LOCAL LAWS AVAILABLE FROM:
- ADAPSO at (703) 522-5055

ROGUE PROGRAM READINGS

especially highly recommended

- Branscomb, Rogue Computer Programs and Computer Rogues – Tailoring the ...
- ... Punishment to Fit the Crime
- Stefanac, Mad Macs
- Spafford, The Internet Worm Incident
- Virus Protection Software: Summary of Features and Performance Tests (PC Mag.)
- VIRUS–L/comp.virus (newsgroup), contact krvw@SEI.CMU.EDU
- VALERT–L, exclusively for posting substantiated virus alerts, krvw@SEI...
- RISKS/comp.risks, RISKS–Request@CSL.SRI.COM
- Zardoz limited to registered site sec. admins., zardoz!neil@uunet.uu.net
- Highland, Computer Virus Handbook, 375 pp., Elsevier Adv. Tech., NY, $153
- Spafford, Heaphy, and Ferbrache, Computer Viruses: Dealing with ...
- ... Electronic Vandalism & Programmed Threats, ADAPSO, approx. $10
- Hoffman, Rogue Programs, Van Nostrand Reinhold, Spring 1990

all in Hoffman, *Rogue Programs*

EASIEST PENETRATION PRINCIPLE

> An intruder must be expected to use any available means of penetration. This will not necessarily be the one against which the most solid defense has been installed.
>
> Charles Pfleeger 1988

TYPICAL ATTACKS

- Attacks on Hardware
- Attacks on Software
- ... Software Deletion
- ... Software Modification
- ... Software Theft, including Attacks on Data
- Data Secrecy
- Data Integrity

ASSETS OF A COMPUTER SYSTEM

- Hardware
- Software
- Data

SECURITY RESPONSIBILITIES

- Security
- Integrity
- Availability

MAJOR THREATS TO A SYSTEM

- Interruption
- Interception
- Modification
- Fabrication

PRINCIPLE OF EFFECTIVENESS

> Controls must be used to be effective. They must be efficient, easy to use, and appropriate.
>
> Pfleeger, 1988

BASIC ENCRYPTION & DECRYPTION

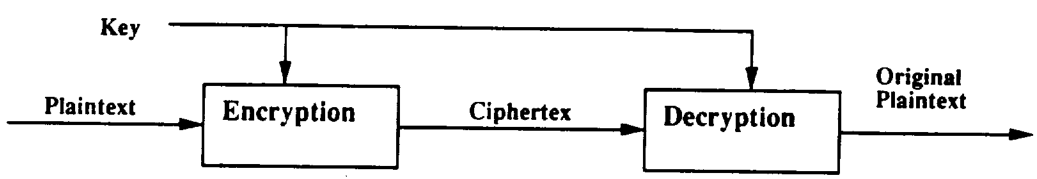
- Encryption: a means of attaining secure communications over insecure channels
- CLASSICAL METHODS FIRST – SUBSTITUTION AND TRANSPOSITION

TERMINOLOGY

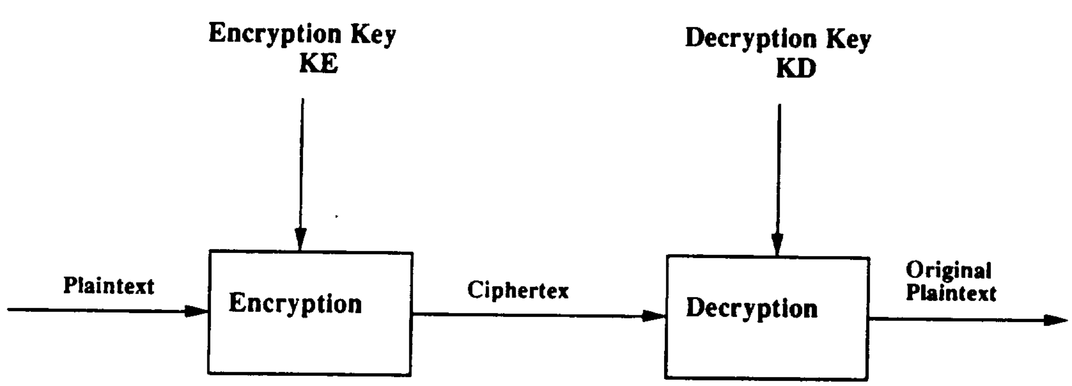
- S: Sender
- R: Receiver
- S sends message to R over transmission medium T
- There can be O, an outsider, who is an interceptor or intruder

TERMINOLOGY

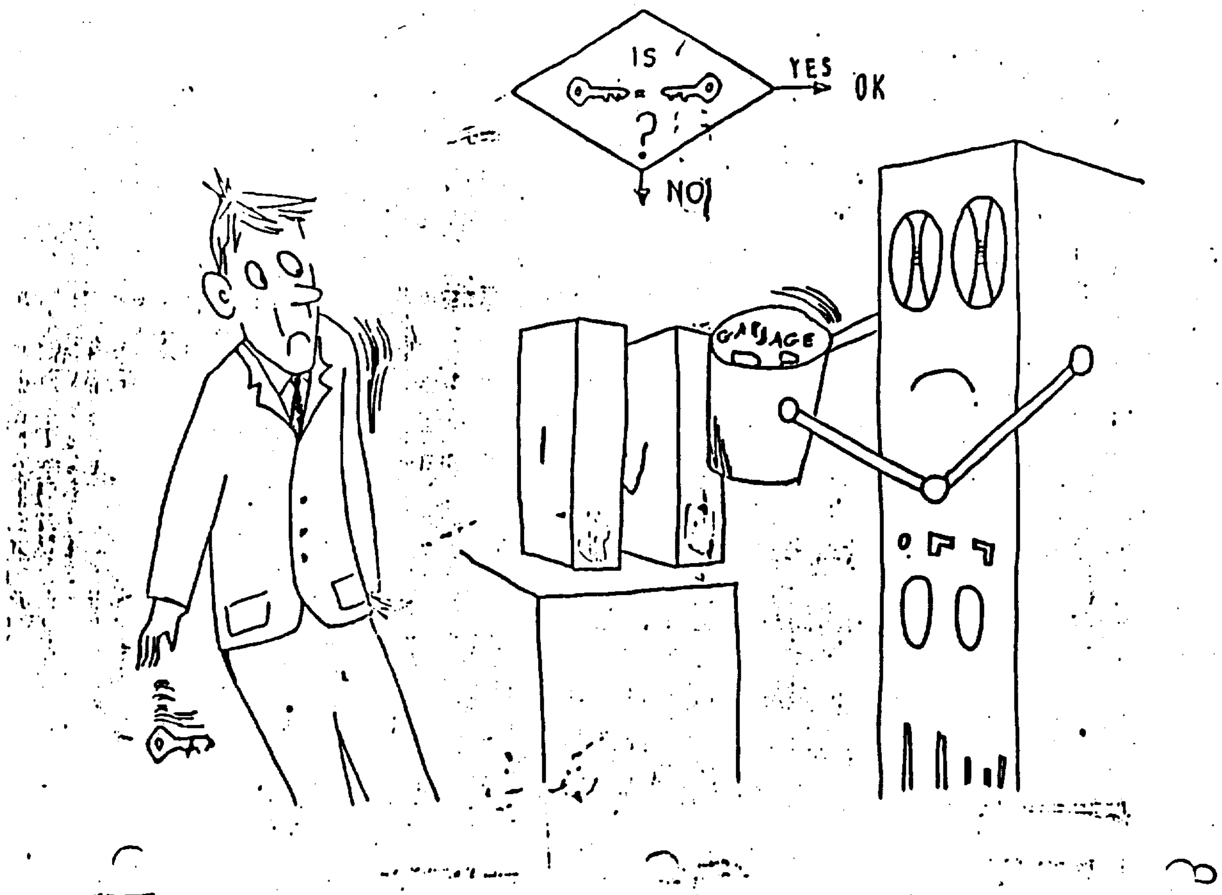
- ENCRYPTION: encoding a message so that its meaning is not obvious
- DECRYPTION: the reverse
- CODE: word(s) into word(s)
- CIPHER: symbols (characters) into other symbols (characters or bit strings)
- ENCRYPTION: covers both encoding and enciphering
- PLAINTEXT: original form of message; P= [p1, p2, ..., pn] P=D(C)
- CIPHERTEXT: encrypted form, C=[c1, c2, ..., cm]. C=E(P)
- P = D(E(P))



Single-Key Cryptosystem



Public-Key Cryptosystem



SUBSTITUTION CIPHERS

Monoalphabetic

- Caesar cipher (Julius Caesar was said to have employed it; many children do)
- $Ci = E(Pi) = Pi + 3$
- Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher: defghijklmnopqrstuvwxyzabc
- Example: TREATY IMPOSSIBLE
wuhdwb lpsrvvlech
- Easy to use in field: algorithm need not be written down.
- Once cracked, it's really cracked!

Example message: plaintext and ciphertext

ENCRYPTION IS A MEANS OF ATTAINING SECURE COMPUTATION
OVER INSECURE CHANNELS
BY USING ENCRYPTION WE DISGUISE THE MESSAGE SO THAT
EVEN IF THE TRANSMISSION IS DIVERTED
THE MESSAGE WILL NOT BE REVEALED

hqfubswlrq lv d phdqv ri dwwdlqlqj vhfxuh frpsxwdwlrq
ryhu lqvhfxuh fkdqqhov
eb xvlqj hqfubswlrq zh glvjxlvh wkh phvvdjh vr wkdw
hyhq li wkh wudqvplvvlrq lv glyhuwhg
wkh phvvdjh zloo qrw eh uhyhdohg

Caesar Cipher, k = 3

COUNTS AND RELATIVE FREQUENCY: example ciphertext

hqfubswlrq lv d phdqv ri dwwdlqlqj vhfxuh frpsxwdwlrq
ryhu lqvhfxuh fkdqqhov
eb xvlqj hqfubswlrq zh glvjxlvh wkh phvvdjh vr wkdw
hyhq li wkh wudqvplvvlrq lv glyhuwhg
wkh phvvdjh zloo qrw eh uhyhdohg

MONOALPHABETIC CIPHERS

Cryptanalysis

- Guessing, using clues: short words, common initial letters, etc.
- Frequency distributions

TABLE 2.2 FREQUENCIES IN EXAMPLE CIPHER.

Letter	Count	Percent	Letter	Count	Percent
a	0	0.00	n	0	0.00
b	3	1.80	o	4	2.41
c	0	0.00	p	5	2.99
d	11	6.59	q	16	9.58
e	2	1.20	r	9	5.39
f	6	3.61	s	3	1.80
g	4	2.40	t	0	0.00
h	26	15.56	u	8	4.79
i	2	1.20	v	17	10.18
j	5	2.99	w	14	8.38
k	5	2.99	x	5	2.99
l	16	9.58	y	4	2.40
m	0	0.00	z	2	1.20
ALL	167				

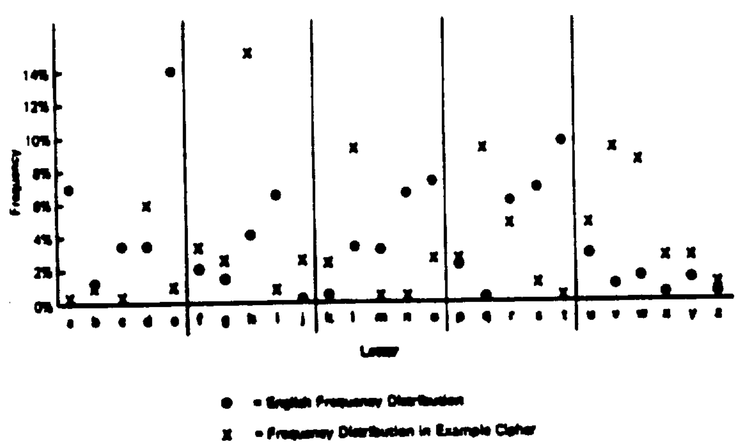


Figure 2.3 Frequencies of Sample Cipher against Normal Text.

(Pfleeger)

TABLE 2.1 LETTER FREQUENCY DISTRIBUTIONS IN ENGLISH AND PASCAL

Letter	English Count	English Percent	Pascal Count	Pascal Percent
a	3312	7.49	664	4.70
b	573	1.29	197	1.39
c	1568	3.54	878	6.22
d	1602	3.62	511	3.61
e	6192	14.00	1921	13.60
f	966	2.18	504	3.57
g	769	1.74	294	2.08
h	1869	4.22	478	3.39
i	2943	6.65	1215	8.60
j	119	0.27	6	0.04
k	206	0.47	87	0.61
l	1579	3.57	722	5.11
m	1500	3.39	270	1.91
n	2982	6.74	1157	8.19
o	3261	7.37	835	5
p	1074	2.43	340	2.41
q	116	0.26	12	0.08
r	2716	6.14	1147	8.12
s	3072	6.95	594	4.21
t	4358	9.85	1311	9.28
u	1329	3.00	377	2.66
v	512	1.16	127	0.89
w	748	1.69	193	1.36
x	123	0.28	139	0.98
y	727	1.64	137	0.96
z	16	0.04	5	0.03

POLYALPHABETIC SUBSTITUTION

Example

- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- adgjmpsvybehknqtwzcfilorux
- PI1(a)= 3a mod 26 above for Odd Positions
- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- nsxchmrwbglqvafkpuzejotydi
- PI2(a)= 5a + 13 mod 26 above for even positions
- TREAT YIMPO SSIBL E
- fumnf dyvtf czysh h
- Note SS -> cz, E -> m or h (T still -> f both times, etc.)

SECURITY OF ENCRYPTION

- Not always what it seems
- Brute force attack on a message may take all 26! decipherments.
- At one decipherment per microsecond, assuming a cryptanalyst with the ...
- patience required to review the probable looking candidate plaintexts, ...
- it would still take over 10,000 YEARS to review all the decipherments.
- But ... frequency analysis really cuts down this time for messages long enough

CRYPTANALYSIS

Methods of Attack

- Attempt to break a single message
- Try to find patterns in encrypted messages and then induct the algorithm.
- Try to find a general weakness in the algorithm itself
- Cryptanalyst relies upon these:
- Encrypted messages
- Known encryption algorithms
- Intercepted plaintext
- Known or suspected plaintext
- Mathematical techniques and tools
- Language properties
- Computers
- Ingenuity and luck

INDEX OF COINCIDENCE

- IC is a measure of variation between frequencies in a distribution
- Let PROBa = probability of an a, etc.
- For a perfectly flat distribution, PROBa=PROBb=...=PROBz=1/26=0.038
- On a graph of a true distribution, a peak is a relative frequency > 0.038
- A valley is a relative frequency < 0.038
- Roughness of distribution of English text against 0.038 as a baseline:

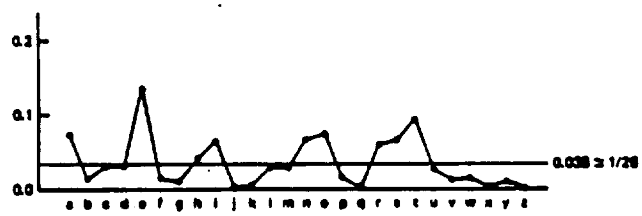


Figure 2.8 Roughness of Distribution of English Text.

- IF we have lots of ciphertext
- AND underlying plaintext has a fairly standard distribution of letters,
- THEN we can use IC to predict the number of alphabets.

INDEX OF COINCIDENCE

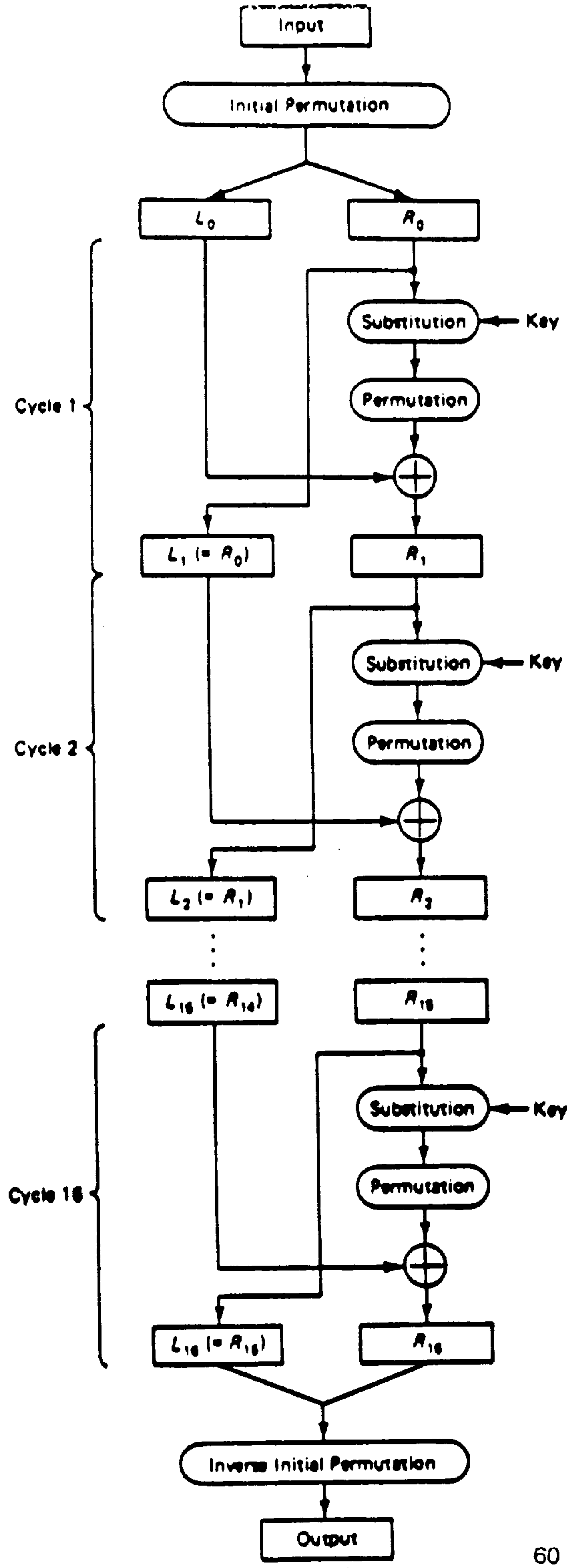
NUMBER OF ALPHABETS	INDEX OF COINCIDENCE
1	.068
2	.052
3	.047
4	.044
5	.044
10	.041
large	.038

CRYPTOGRAPHY

Further References

- KAH67 David Kahn, *The Codebreakers:* classic beside reading
- FRI76 Friedman: original work first part of 20th century; key in WW1, WW2
- SIN66 Sinkov: highly readable presentation of elementary crypto
- KON80: Konehim: more mathematical
- MEY82: Meyer and Matyas: more mathematical
- DEN : Denning: basic intro incl intro to info theory concepts

DES SINGLE KEY SYSTEM



PUBLIC KEY SYSTEMS

Originators: Diffie & Hellman 1976

- Two keys: public and private
- User's public key not secret; private key is secret
- Saves on total number of keys to manage, since n–user system needs n(n–1)/2 keys
- Unreasonable to expect users to memorize that many keys
- EXISTING USERS   NEW USER

- NEW KEYS ADDED DENOTED BY – – –

- Keys operate as inverses
- P = D(Kpriv,E(Kpub,P))
- P = D(Kpub,E(Kpriv,P))
- Now n users require only 2n keys

R S A

Creating the public key

1. Pick an odd number, E	E=5
2. Pick two prime numbers, P and Q, where (P-1)(Q-1)-1 is evenly divisible by E	ideally both P and Q are about 100 digits P=7, Q=17
3. Multiply P and Q to get N (about 200 digits	N=PxQ=7x17=119
4. Concatenate N and E to get the encrypting or public key	Public key=NE=1195

Creating the private key

1. Subtract 1 from P, Q, and E, multiply the results, and add 1	(P-1)(Q-1)(E-1)+1=6x16x4+1=385
2. Divide result by E to get D	D=385/5=77
3. Concatenate N and D to get the decrypting or private key	Private key=ND=11977

Encrypting the message with the public key

1. The message is converted to numerical equivalents. The letter S, for example, may be represented by 19	Plain text=19
2. The algorithm $C=P^e \bmod N$	
a) raise plain text to power of E	19^5=2476099
b) divide by N	2476099/119=20807 with a remainder of 66
3. The remainder is the encrypted value or cipher text	Cipher text=66

Decrypting the cipher text with the private key

1. The algorithm	
a) raise cipher text to power of D	66^{77}=1.27...E140
b) Divide by N	1.27...E140/119=1.069...E138 with a remainder of 19
2. The remainder is the decrypted value or plain text	Plain text=19

SINGLE KEY SYSTEMS

- If key remains secret, authentication is provided also --

- Only the legitimate user can produce a message that will decrypt properly

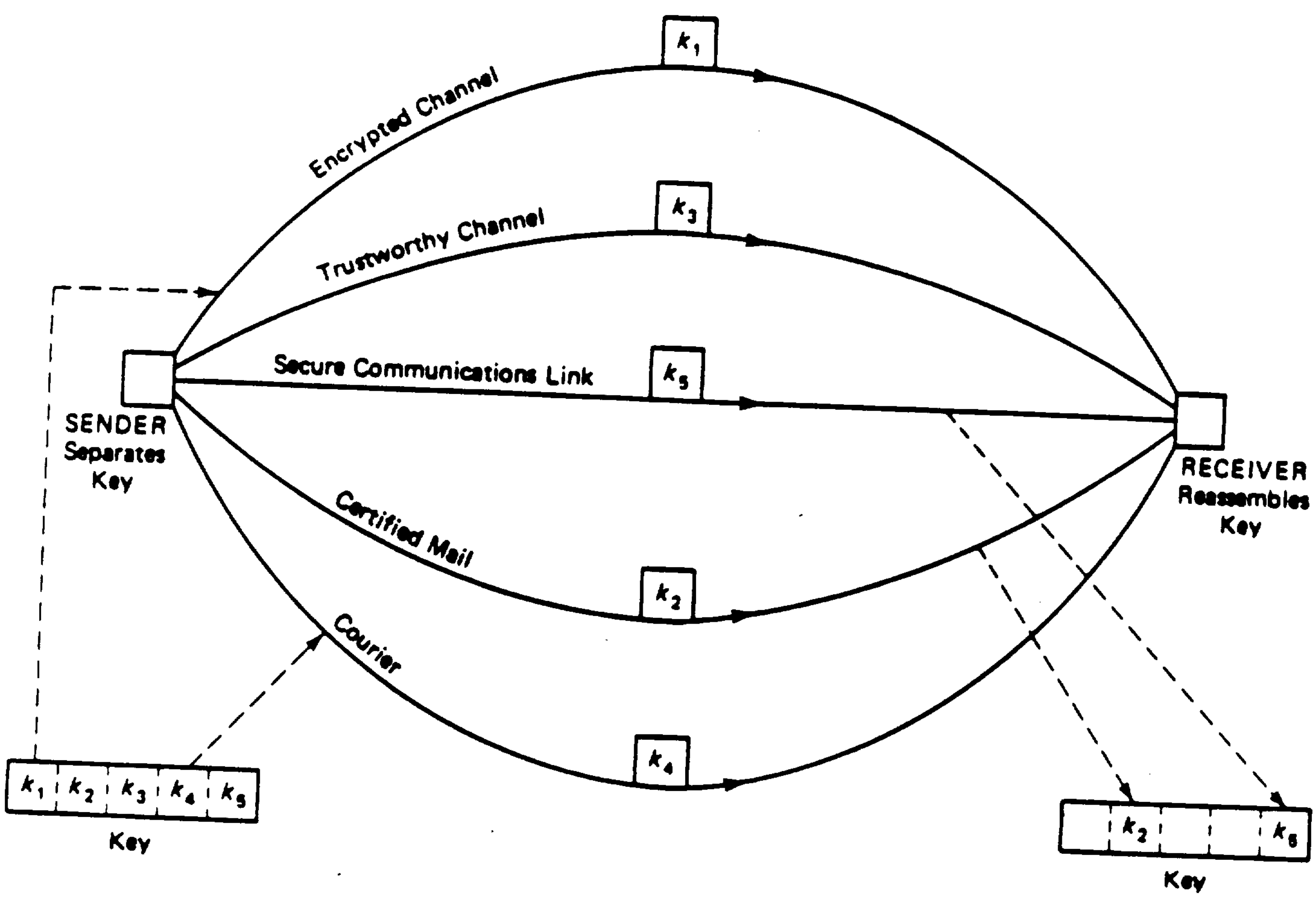
SINGLE KEY SYSTEMS - PROBLEMS

- If key revealed, interceptors can immediately decrypt all available information

- Imposter can produce and send bogus messages

- (Change keys fairly frequently)

- Key distribution problem -- separate channels, by hand, in pieces (Pfleeger)



PROTECTIONS FOR O.S. USERS

- Operating systems support multiprogramming, and thus provide...
- – memory protection
- – file protection
- – control of access to objects
- – user authentication
- (in non–PC systems)

Objects Which Require Security

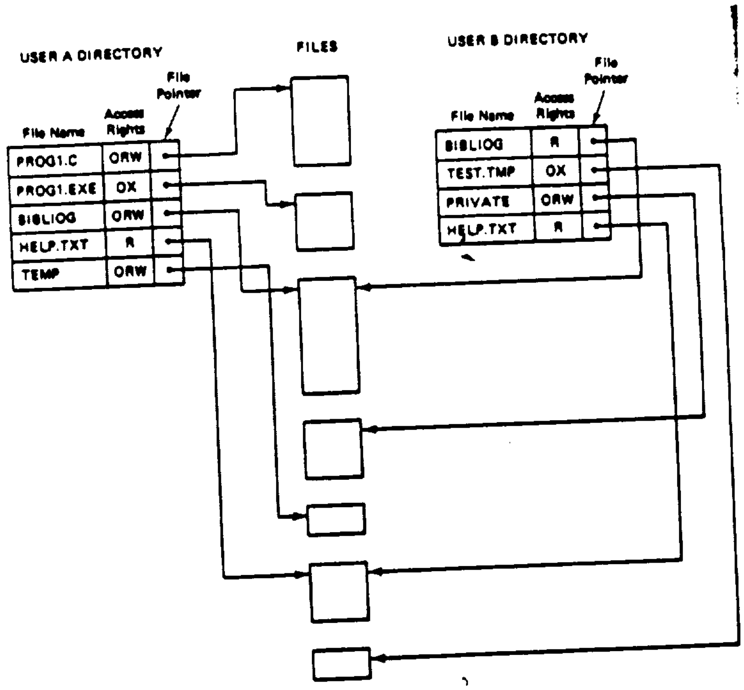
- – memory
- – sharable I/O devices, e.g., disks
- – serially reusable I/O devices, e.g., printers and tape drives
- – sharable programs and subprocedures
- – sharable data

LEVELS OF SEC. OFFERED BY O.S.

- *No protection.* OK when sensitive procedures run at separate times.
- *Isolation.* Each process has own address space, files, and other objects.
- *Share all or share nothing.* Everything is either public or private to owner.
- *Share via access limitation.* Whether user can access object is on a list.
- *Share by capabilities.* Dynamic creation of sharing rights for objects.
- *Limit use of an object.* Protects use as well as access (e.g., view, don't print)
- GRANULARITY: bit, byte, element/word, field, file, volume? It depends!

MECHANISMS FOR O.S. SECURITY

- Physical separation (e.g., separate printer for different output sec levels)
- Temporal separation (periods processing)
- Logical separation (O.S. protects objects outside user pgm's domain)
- Cryptographic separation (data and computations unintelligible to others)
- Combinations of the above



PROTECTION MECHANISMS

Access Control Matrix

- A matrix where each row is a subject, each column an object, and the ...
- ... matrix entry specifies the rights.
- Usually sparse, thus infrequently used as a matrix.
-

Subjects \ Objects	BIBLIOG	TEMP	F	HELP.TXT	C_COMPILER	LINKER	SYS_CLOCK	PRINTER
User_A	ORW	ORW	ORW	R	X	X	R	W
User_B	R	–	–	R	X	X	R	W
User_S	RW		R	R	X	X	R	W
User_T	–	–	–	R	X	X	R	W
Sys MGR	–	–	–	RW	OX	OX	ORW	O
USER_Svcs	–	–	–	O	X	X	R	W

FILE PROTECTION MECHANISMS

- Additional mechanisms came to include a password associated with a file.
- Problems with passwords include:
- – Loss. System administrator could intervene to fix.
- – Disclosure. If PW then changed, all legitimate users must be notified.
- – Revocation of a right ==> PW is changed==> same problems as disclosure

USER AUTHENTICATION

- Most authentication systems for computers use something user knows
- ... like a password. (Can also use something user has, or is)
- Passwords: length and format vary and can greatly influence security
- Identification <> authentication <> authorization
- Generally system requires authentication AND identification to match

DISTRIBUTION OF ACTUAL PW's

(Morris and Thompson, 1979)

- 86% of a sample of 3289 could be uncovered in one week, at 1 ms/password
- 0.5% were a single ASCII character
- 2% were two ASCII characters
- 14% were three ASCII characters
- 14% were four letters
- 21% were 5 letters, all the same case
- 18% were 6 lower case letters
- 15% were words in dictionaries or lists of names
- (The above total to 86%)

ADDITIONAL AUTHENTICATION INFO

- Can only allow users access at certain times and/or from certain terminals
- Problem if users work overtime or need access from out of town
- Can be solved by prearrangement with security office, but still a hassle

THINK OF A WORD.

- Is it long?
- Is it uncommon?
- Is it hard to spell or pronounce?
- -- Probably "NO" to all of the above.
- If the chosen password is too short, search space dramatically falls, e.g.,
- All 5 character passwords take only 12,356 seconds ~ 3.5 hrs. @1 per ms.
- Typical chosen passwords: names of friends or family, projects, etc.

ATTACKS ON PASSWORDS

- Can try all combinations of n characters or n password-characters
- Can try likely passwords for the user *(a priori* knowledge)
- Can search for an unencrypted system password list
- Can spoof the user

CERTIFICATION OF SECURE OSs

- CERTIFICATION is process of assessing quality of the testing that has been...
- ... performed and assigning a measure of confidence in the correctness of the...
- ... system.

METHODS OF EVALUATION

of an Operating System

- formal verification
- informal validation
- penetration analysis

EVALUATING OSs: FORMAL VERIF.

- Most precise method of analyzing security.
- The OS is reduced to a "theorem" which is then proven.
- The thm. asserts the OS is correct: it does what is should and nothing else.
- CAN TAKE MANY PERSON-YEARS OF EFFORT.
- Computer programs ("theorem provers") help.
- Example:

HONEYWELL INFORMATION SYSTEMS, INC.

Federal Systems Division

SCOMP Trusted Software

Formal Specifications

5. File Display

```
Proof of MARKED_CORRECTLY:

(1)   H1: DISPLAY_FILE_ACTION (FDISP, PATH, ST)
      H2: NULL (DISPLAY) ne FDISP
  ->
      C1: PAGES_LABELED (FDISP, PATH, ST)

Expanding DISPLAY_FILE_ACTION results in the goal

(3)   H1:     READABLE_FILE (PATH, ST) & SEGMENTS_CONSISTENT (PATH, ST)
        ->     CONCAT_ALL_PAGES (FDISP)
             = GET_FILE_CONTENTS (ST.FILESYS.ROOTS[PATH].FIL)
           & PAGES_LABELED (FDISP, PATH, ST)
      H2:        not READABLE_FILE (PATH, ST)
           or not SEGMENTS_CONSISTENT (PATH, ST)
        -> NULL (DISPLAY) = FDISP
      H3: NULL (DISPLAY) ne FDISP
  ->
      C1: PAGES_LABELED (FDISP, PATH, ST)

Simplifying, we get:

(5)   H1: READABLE_FILE (PATH, ST)
      H2: SEGMENTS_CONSISTENT (PATH, ST)
      H3:     READABLE_FILE (PATH, ST) & SEGMENTS_CONSISTENT (PATH, ST)
        ->     CONCAT_ALL_PAGES (FDISP)
             = GET_FILE_CONTENTS (ST.FILESYS.ROOTS[PATH].FIL)
           & PAGES_LABELED (FDISP, PATH, ST)
      H4: NULL (DISPLAY) ne FDISP
  ->
      C1: PAGES_LABELED (FDISP, PATH, ST)

Simplifying, we get:

(7)   C1: TRUE


This completes the proof of MARKED_CORRECTLY.

          * * * * * * * * * * * * * * * * * * * * * * * * *

Proof of PRINTED_CORRECTLY:

(1)   H1: DISPLAY_FILE_ACTION (FDISP, PATH, ST)
      H2: NULL (DISPLAY) ne FDISP
```

THE LIMITS TO MODELS

> "...if the model grows to the point where it can no longer be easily understood, then much of its value is lost." -- "Proving Multilevel Security of a System Design", Proc. 6th ACM Symposium on Operating Systems Principles (Nov. 1977), 57-65.
>
> R. J. Feiertag, et al.

PROBLEMS WITH FORMAL VERIF.

- Note how the algorithm in the flowchart is often used in intro programming,
- and it's easy to convince yourself it is correct.
- The algorithm ABOUT THE ALGORITHM (the verification) takes longer to explain,
- is longer to write, and is tougher to understand.
- This illustrates the two principal difficulties with formal verification:
- time: time-consuimg to state the assertions at each step and verify flow
- complexity: for some large systems it is hopeless (spaghetti code)

PROBLEMS WITH PROGRAM PROOFS

- Formal proofs for SCOMP are inches high. Who reads and, then, who believes?
- DeMillo, Lipton, Perlis., "Social processes and proofs of theorems ...
- and programs", CACM *22*, 5 (May 1979), 271–280.
- Responding letters by van den Bos, Lamport, and Maurer, CACM *22*, 11 (Nov. 1979)
- Fetzer, J., "Program verification: the very idea", CACM *31*, 9 (Sep. 1988)
- Angry responses in CACM (March 1989?)

CAN NON-TOY PROOFS BE DONE?

"The problem with engineers is that they tend to cheat in order to get results. The problem with mathematicians is that they tend to work on toy problems in order to get results. The problem with program verifiers is that they tend to cheat at toy problems in order to get results." -- 1980 IEEE Symp. on Sec. and Pri., 145ff.

S.R. Ames Jr. & J.G. Keeton-Williams

Tiger Team Penetration Testing

- Team of "experts" tries to break the system being tested
- An OS that fails such a test is known to have flaws. One that passes is...
- not guaranteed to be error–free.

NCSC CERTIFICATION

"The Orange Book"

- *Trusted Computer System Evaluation Criteria*
- Originally, idea was to have something to stick on back of an RFP.
- Progression of security requirements reflected in rating:
- A1
- B3
- B2
- B1
- C2
- C1
- D

Evaluation Criteria For Trusted Computer Systems

D ⟶ C1 ⟶ C2 ⟶ B1 ⟶ B2 ⟶ B3 ⟶ A1 - - → A2

Security Mechanisms ⟶ Secure System Design

Informal Analysis ⟶ Formal Analysis

Assurance by Test ⟶ Assurance By Formal Proof

Minimal Protection ⟶ Mathematically Proven Protection

Least Desirable ⟶ Most Desirable

prc

TRUSTED CPTR SYS EVAL CRITERIA

(TCSEC)

Criteria	Requirement D	C1	C2	B1	B2	B3	A1
Security Policy							
Discretionary Access Control	■	●	●	→	→	●	→
Object Reuse	■	■	●	→	→	→	→
Labels	■	■	■	●	●	→	→
Label Integrity	■	■	■	●	→	→	→
Exportation of Labeled Information	■	■	■	●	→	→	→
Labeling Human-Readable Output	■	■	■	●	→	→	→
Mandatory Access Control	■	■	■	●	●	→	→
Subject Sensitivity Labels	■	■	■	■	●	→	→
Device Labels	■	■	■	■	●	→	→
Accountability							
Identification and Authentication	■	●	●	●	→	→	→
Audit	■	■	●	●	●	●	→
Trusted Path	■	■	■	■	●	●	→
Assurance							
System Architecture	■	●	●	●	●	●	→
System Integrity	■	●	→	→	→	→	→
Security Testing	■	●	●	●	●	●	●
Design Specification and Verification	■	■	■	●	●	●	●
Covert Channel Analysis	■	■	■	■	●	●	●
Trusted Facility Management	■	■	■	■	●	●	→
Configuration Management	■	■	■	■	●	→	●
Trusted Recovery	■	■	■	■	■	●	→
Trusted Distribution	■	■	■	■	■	■	●
Documentation							
Security Features User's Guide	■	●	→	→	→	→	→
Trusted Facility Manual	■	●	●	●	●	●	→
Test Documentation	■	●	→	→	●	→	●
Design Documentation	■	●	→	●	●	●	●

Legend: ■: no requirement; ●: additional requirement;
→: same requirement as previous class

HIGHLIGHTS OF TCSEC

- D: Zilch
- C1: Discretionary Security Protection
- -- for cooperating users processing data of same sensitivity level
- -- Users allowed to protect their own data. Security fcns protected.
- -- Example: MVS running RACF
- C2: Controlled Access Protection
- -- Protection implementable down to a single user.
- -- Audit trail can track each individual's access to each object
- -- No residue exposure
- -- Examples: MVS with ACF2, DEC VAX VMS

- B1: Labeled security protection
- Nondiscretionary (mandatory) access control of each subject and object.
- Access control based on a model with both hierarchical and other categories
- Mandatory access policy: Bell-LaPadula model
- Design documentation, source and object code thoroughly analyzed and tested
- An "informal or formal model" of the security policy shall be available

TCSEC (Continued)

- B2: Structured Protection
- Major enhancement for B2 is design requirement:
- A verifiable top-level design, and testing must confirm that system ...
- ... implements this design
- Modular
- Least privilege
- Access control policies enforced on all subjects and objects, including devices
- Covert channel analysis required
- System protected against external interference or tampering
- Example: Honeywell MULTICS
- B3: Security domains
- High-level design must be complete and conceptually simple
- Convincing argument must exist that system implements this design
- Small enough for extensive testing
- Design shall make significant use of layering, abstraction, info. hiding
- Complete mediation
- Security functions tamperproof
- Highly resistant to penetration
- System audit facility can identify when a violation of security is imminent

TCSEC The Last

- A1: Verified Design
- Formally verified system design
- System capabilities same as class B3
- Five criteria for A1 certification:
- -- formal model of protection system, proof of its consistency and adequacy
- -- formal top level specification of protection system
- -- demonstration that top level specification conforms to model
- -- implementation "informally" shown consistent with the specification
- -- formal analysis of covert channels
- Example: Honeywell SCOMP

EXAMPLES OF SECURITY IN OSs

Unix

- never intended to have high security
- easy sharing much more important
- Unix sysadmin is by dogma part-time and a programmer with few security functions
- One "superuser" who can do anything
- Most system attacks aim at obtaining rights of superuser
- Most sensitive utility programs are OWNED by SUPERUSER.
- Using setuid, user's access rights are those of owner of utility, not user.
- One security flaw in one utility program gives very wide access.
- (Note SENDMAIL flaw under Unix in Internet Worm incident of 11/88)

VAX/VMS

- Started out with moderate protection, mainly discr. controls by users
- Modified, so now approved at C2 TCSEC level. Modifications include
- *access controls* at single subject/single object level
- password controls
- *auditing functions* which track selected security events
- *monitoring functions* that warn administrators of suspicious events
- *encryption* available at user's request

MANDATORY AND DISCRETIONARY

- Rings, as basically used, are nondiscretionary or *mandatory* controls:
- They apply to all objects, regardless of contents or owner.
- A given segment can also contain *discretionary* controls to further check

TYPICAL FLAWS IN OPER SYSTEMS

- I/O Processing is a big weak spot.
- -- *Independent*, intelligent devices, controllers, and channels operating
- -- These independent units fall outside the kernel or the OS's security code
- -- I/O code can be more complex; so harder to review or prove
- -- I/O code sometimes bypasses OS functions for efficiency => bypass check

- Ambiguity in access policy
- -- Important to have separation of users/programs and protection/isolation
- -- Also very useful to have sharing

TYPICAL FLAWS IN OPER SYSTEMS

Continued

- Incomplete mediation
- -- Without explicit reqmt, designers minimize machine resources used

- Generality
- -- Some add-on packages must take on same access privileges as oper system
- -- The "hooks" provided are trapdoors for any user to penetrate the oper sys

EXAMPLES OF O.S. FLAWS

- I/O commands often reside in user memory space. Any user can alter ...
- ... source or dest addr after I/O command has started.
- (works this way because complete mediation may be too costly)

- SVC for installation of other security packages. When invoked, returned to...
- ... user in privileged mode. No add'l checks to authenticate pgm invoking SVC

CERTIFICATION OF SECURE OSs

- CERTIFICATION is process of assessing quality of the testing that has been...
- ... performed and assigning a measure of confidence in the correctness of the...
- ... system.

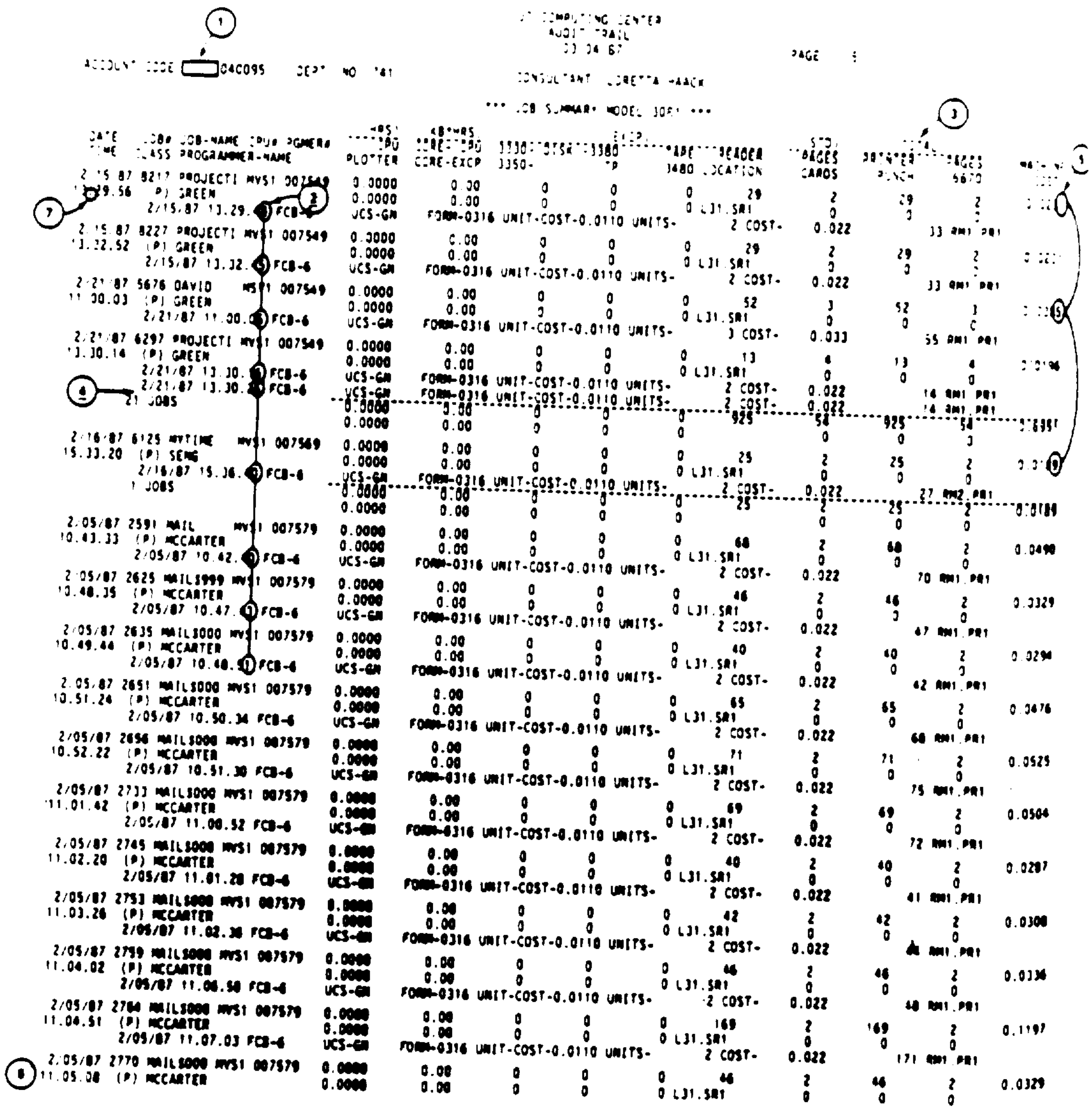
METHODS OF EVALUATION

of an Operating System

- formal verification
- informal validation
- penetration analysis

COVERT CHANNELS

Hidden means to communicate info.



- (Figure 5.3 from Pfleeger, *Security in Computing*)

(small amounts only)

DENIAL OF SERVICE ATTACKS

A. Greedy Programs
   - Accidentally or intentionally consume all resources
   - Ex: Computer PI in background

B. Loops
   - I/O (channel) programs which never terminate, thus halt CPU

C. Viruses

DATA BASE ADVANTAGES

- SHARED ACCESS to one common, centralized set of data
- MINIMIAL REDUNDANCY, so individual users neet not collect/maintain own data
- DATA CONSISTENCY: a change to one value affects all users' views
- DATA INTEGRITY: data values *more* secure against accidental/malicious changes
- CONTROLLED ACCESS: only authorized users allowed to view or modify data values
- EFFICIENT (hopefully).

EXAMPLE OF NONCLUSIVE PROPERTIES USED TO NARROW DOWN A FIELD

P1: MATH DEGREE FROM CARNEGIE TECH

P2: PH. D. IN COMPUTER SCIENCE FROM STANFORD

P3: CURRENTLY ON FACULTY AT GWU

P4: (THAT'S ENOUGH!)

C(ALL)=17 000 000 STUDENTS, FACULTY, STAFF

C(P1) = 7 000

C(P2) = 100

C(P3) = 1300

C(P1 AND P2 AND P3) = 1 (YOURS TRULY!)

C(P1 AND P2 AND P3 AND P4) = 1 OR 0, DEPENDING ON P4.

STATISTICAL DATA BANKS

METHODS FOR DOSSIER EXTRACTION

1. ISOLATE THE INDIVIDUAL IN THE DATA BANK.
2. PLACE THE INDIVIDUAL IN A GROUP WITH A GIVEN PROPERTY.
3. PLACE THE INDIVIDUAL OUTSIDE A GROUP WITH A GIVEN PROPERTY.
4. ADD DUMMY ENTRIES TO SUBVERT MINIMUM REPORTABLE COUNTS.
5. COMPROMISE BY SIMULTANEOUS EQUATIONS.
6. TRACKERS.
7. DOUBLE TRACKERS.
8. IMPLIED QUERIES.

2. PLACE THE INDIVIDUAL IN A GROUP WITH A GIVEN PROPERTY

IF WE FIND N OF MR. X'S PROPERTIES SUCH THAT

$C(P_1 \& P_2 \ldots \& P_N) = C(P_1 \& P_2 \& \ldots \& P_N \& P_0)$

THEN HE ALSO POSSESSES PROPERTY P_0.

3. PLACE THE INDIVIDUAL OUTSIDE A GROUP WITH A GIVEN PROPERTY

$C(P^*)$ = NO. OF PEOPLE (EXCEPT MR. X) WITH PROPERTY P_0. $(P^* = \bar{P}_1 \vee \bar{P}_2 \ldots \vee \bar{P}_N) \wedge P_0$

$C(P_0)$ = NO. OF PEOPLE WITH PROPERTY P_0.

<u>ASSUMING</u> MR. X UNIQUELY IDENTIFIED BY $P_1, P_2, \ldots, P_N$,

THEN IF $C(P_0) = C(P^*)$, HE DOES NOT HAVE P_0, OTHERWISE HE DOES

4. ADD DUMMY ENTRIES TO SUBVERT MINIMUM REPORTABLE COUNTS

5. COMPROMISE BY SIMULTANEOUS EQUATIONS

DATA BASE:

Contributor	Business Area	Political Leaning	Favoritism Shown by Administration	Geographic Area
C1	Steel	Democrat	High	Northeast
C2	Steel	Republican	Medium	West
C3	Steel	Independent	Low	South
C4	Sugar	Democrat	Medium	Northeast
C5	Sugar	Republican	Low	Northeast
C6	Sugar	Independent	High	West
C7	Oil	Democrat	Low	South
C8	Oil	Republican	High	South
C9	Oil	Independent	Medium	West

ONLY DATA OBTAINABLE IS SUM GIVEN BY ALL CONTRIBUTORS SHARING A COMMON ATTRIBUTE

<u>EXAMPLES</u>: CONTRIBUTIONS FROM STEEL INDUSTRY C1+C2+C3

CONTRIBUTIONS FROM REPUBLICANS C2+C5+C8

ETC.

USING THESE ONLY WE CAN OBTAIN

SOLVE 12 EQNS. IN 9 UNKNOWNS TO FIND C1, C2,... C9

		Contributing Group	Amount
C1 + C2 + C3	270 000	Steel (C1+C2+C3)	270 000
C4 + C5 + C6	120 000	Sugar	120 000
C7 + C8 + C9	540 000	Oil	540 000
C1 + C4 + C7	186 000	Democrats	186 000
C2 + C5 + C8	564 000	Republicans	564 000
C3 + C6 + C9	180 000	Independents	180 000
C1 + C6 + C8	510 000	High favoritism	510 000
C3 + C5 + C7	174 000	Low favoritism	174 000
C2 + C4 + C9	246 000	Medium favoritism	246 000
C1 + C4 + C5	90 000	Northeast	90 000
C2 + C6 + C9	330 000	West	330 000
C3 + C7 + C8	510 000	South	510 000

Implied Query Sets

- Implied query sets of a*b are a*not b and not a * b
- SUMMARY: A query q(a*b) or q(a+b) OK if answerable in [n,N−n].
- Implied queries are a*b, a*not b, not a*b and not a * not b

Figure 6.14 shows the eight implied query sets for the case $m = 3$. The formulas relating a statistic q computed over one of the query sets to the remaining query sets are:

$$
\begin{aligned}
q(a \cdot b \cdot {\sim}c) &= q(a \cdot b) - q(a \cdot b \cdot c) \\
q(a \cdot {\sim}b \cdot c) &= q(a \cdot c) - q(a \cdot b \cdot c) \\
q({\sim}a \cdot b \cdot c) &= q(b \cdot c) - q(a \cdot b \cdot c) \\
q(a \cdot {\sim}b \cdot {\sim}c) &= q(a \cdot {\sim}b) - q(a \cdot {\sim}b \cdot c) \\
&= q(a) - q(a \cdot b) - q(a \cdot c) + q(a \cdot b \cdot c) \\
q({\sim}a \cdot {\sim}b \cdot c) &= q(c) - q(a \cdot c) - q(b \cdot c) + q(a \cdot b \cdot c) \\
q({\sim}a \cdot b \cdot {\sim}c) &= q(b) - q(a \cdot b) - q(b \cdot c) + q(a \cdot b \cdot c) \\
q({\sim}a \cdot {\sim}b \cdot {\sim}c) &= q(All) - q(a) - q(b) - q(c) \\
&\quad + q(a \cdot b) + q(a \cdot c) + q(b \cdot c) - q(a \cdot b \cdot c)
\end{aligned}
$$

TRACKERS

A TRACKER IS A SET OF AUXILIARY CHARACTERISTICS ADDED TO THE ORIGINAL QUERY CHARACTERISTICS TO CREATE ANSWERABLE QUERIES; THE QUESTIONER SUBTRACTS OUT THE EFFECT OF THE AUXILIARY CHARACTERISTICS TO DETERMINE THE ANSWER TO THE QUERY FOR THE ORIGINAL CHARACTERISTICS.

REF: A FAST PROCEDURE FOR FINDING A TRACKER IN A STATISTICAL DATABASE, D. DENNING AND J. SCHLORER

ACM TODS 5, 1 (MARCH 1980) PP. 88-107.

CONSTRUCTING TRACKERS

1. TRACKERS CAN ALMOST ALWAYS BE CONSTRUCTED (PROCEDURE GIVEN IN DENNING AND SCHLORER, "A FAST PROCEDURE FOR FINDING A TRACKER IN A STATISTICAL DATABASE")

2. IN CASES WHERE TRACKERS CAN BE FOUND,
   A. OFTEN, ONLY ONE OR TWO QUERIES ARE NECESSARY;
   B. AT MOST $O(\log_2 N)$ QUERIES ARE NECESSARY, WHERE N IS THE NUMBER OF DISTINCT RECORDS.

3. ANY ATTEMPT TO DETECT THE CONSTRUCTION OF A TRACKER IS LIKELY TO FAIL.

4. IN DATABASES WHERE MOST INDIVIDUALS ARE UNIQUELY IDENTIFIED, COMPROMISE BY TRACKERS IS A VERY SERIOUS THREAT, UNLESS STEPS SUCH AS RANDOM SAMPLE QUERIES ARE USED TO PREVENT COMPROMISE.

STATISTICAL DATA BANKS

General Protective Measures

- **Limitations on Responses Honing in on One (or a Few) Persons**
- **Cell Suppression**
- **Limiting Excessive Overlap Among Queries**
- **Noise: inoculation, output perturbation, data distortion**
- **Limit data bank size (don't create it if N < K)**
- **Sampling**
- **Link files**
- **Random sample queries (powerful against trackers)**
- **Logging**

EXAMPLE OF CELL SUPPRESSION

Student counts by SEX and CLASS

CLASS

SEX	1978	1979	1980	1981	SUM	
FEMALE	1	2	2	1	6	
MALE	3	2	0	2	7	
SUM	4	4	2	3	13	TOTAL

CELL SUPPRESSION

In CELL SUPPRESSION, all sensitive statistics and some non-sensitive statistics are suppressed from tables. Such non-sensitive statistics are called COMPLEMENTARY SUPPRESSIONS.

Example:

CLASS

SEX	1978	1979	1980	1981	SUM	
FEMALE	-	1330	1120	-	3750	
MALE	1930	1150	0	1180	426C	
SUM	2730	2480	1120	1680	8010	TOTAL

Entries in row 1, columns 1 and 4 are suppressed since they represent individual contribution.

Can compute the missing entries by using the sums. Thus, need to also suppress some non-sensitive statistics.

CELL SUPPRESSION (CONTINUED)

	CLASS				
SEX	1978	1979	1980	1981	SUM
FEMALE	-	1330	1120	-	3750
MALE	-	1150	0	-	4260
SUM	2730	2480	1120	1680	8010 TOTAL

Complementary suppression of row 2, columns 1 and 4 yields secure data table.

Random sample queries (RSQs)

- RSQs control compromise by reducing questioner's ability to interrogate the
- ... desired query sets precisely.
- Compromise possible w/ small query sets unless p small or min-query-set-size OK
- Trackers are no longer a useful tool for compromise.
- Relative frequency and average expected values are the true ones.
- Manual attacks with lots of queries infeasible; not so computer-aided

(Denning 1982)

Random Sample Query Control

- Given query q(c), the query processor examines each record i in C.
- It applies a selection function f(C,i) to determine whether i is used in stat
- Set of selected records forms the sampled query set...
- C* = {i in C | f(C,i)=1 } from which query processor returns q*=q(C*)
- A parameter p selects sampling probability that a record is selected.
- The uncertainty introduced is the same as that in sampling entire d.b. with p
- Expected size of a random sample over entire d.b. of size N is pN.

PROTECTING AGAINST STATISTICAL INFERENCE USING A LINK FILE SYSTEM

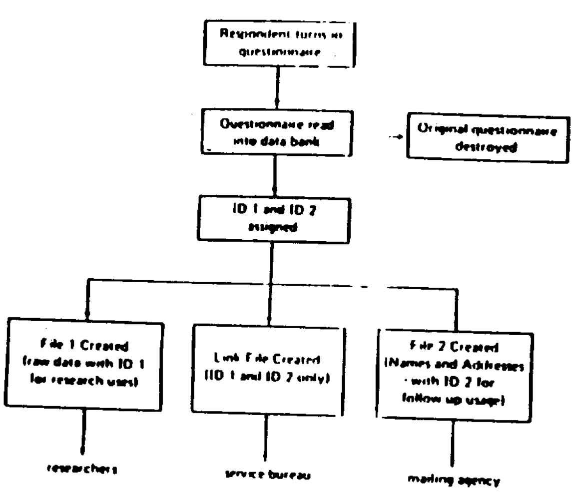


Figure 10-1 Overview of a link file system.

TABLE 10-1 Files Maintained When a Link File System Is Used

File	Contents	Kept by	Used for
File 1	Original Raw Data, ID 1	Researcher	Research
File 2	Original Names and Addresses, ID2	Mailing Agency	Sending Follow-ups
Link File	ID 1 and ID 2	Service Bureau	Matching Responses with Respondents while Preserving Anonymity
New File 1	New Raw Data, ID 1	Researcher	Research
New File 2	New Names and Addresses, ID 2	Mailing Agency	Sending Further Follow-ups

TABLE 10-2 Example of the Contents of Files in a Link File System

Link File		File 1—ID 1 with raw data			File 2—ID 2 with name and address	
ID 1	ID 2	2459	1930-1947	Peoria	1000	John Smith, 54 W 168 St., New York NY., USA
2459	1000		1947-1951	Springfield	2000	Pierre Renault, 30 Cours de la Liberation 75000 Paris, FRANCE
3858	2000		1951-1953	Korea	3000	Anais Opnn, Via Dolorosa, Beirut, LEBANON
7725	3000		1953-1958	Chicago		
			1958-1969	St. Louis		
			1969-	New York		
		3858	1943-1945	Vichy		
			1945-	Paris		
		7725	1950-1953	Port Said		
			1953-1967	Sharm-al-Sheik		
			1967-1970	(Unknown)		
			1970-1975	Beirut		
			1975-	(Unknown)		

User Authectication

- **Exchange of Secrets Protocol**
  - shared encryption key
    - key plus message and password are used for authentication
  - systems do not share an encryption key
    - central authority protocol is used
    - involves a third party which shares an encryption key with both parties.

- **Passphrases**
  - longer version of a password
  - takes more computer memory to store
  - can be used in a challenge-response system
  - examples: line from a song, or a list of countries.

- **Token or Smart Card**
  - Token : "magnetic stripe credit card"
  - Smart Card : embedded micrprocessor

- **Personal Characteristics**
  - fingerprints, pronunciation, and patterns of the retina of the eye.

Data Integrity

- **More Sophisticated Error Codes**
  - permit detection of errors in two or more bits.

- **Digital Signatures**
  - certify the authenticity of a set of data.

- **Notarization**
  - attest to the authenticity of a message.

LAN Topology

- Ring Network
  - Each message is seen by the other nodes
  - One node can deny service to another by withholding or failing to forward messages
  - No central authority can analyze traffic flow in order to detect covert channels

- Other Security in LANs
  - Connectability of LANs increases security risks
  - File Server for an LAN is typically vulnerable to attack, partically when off-line

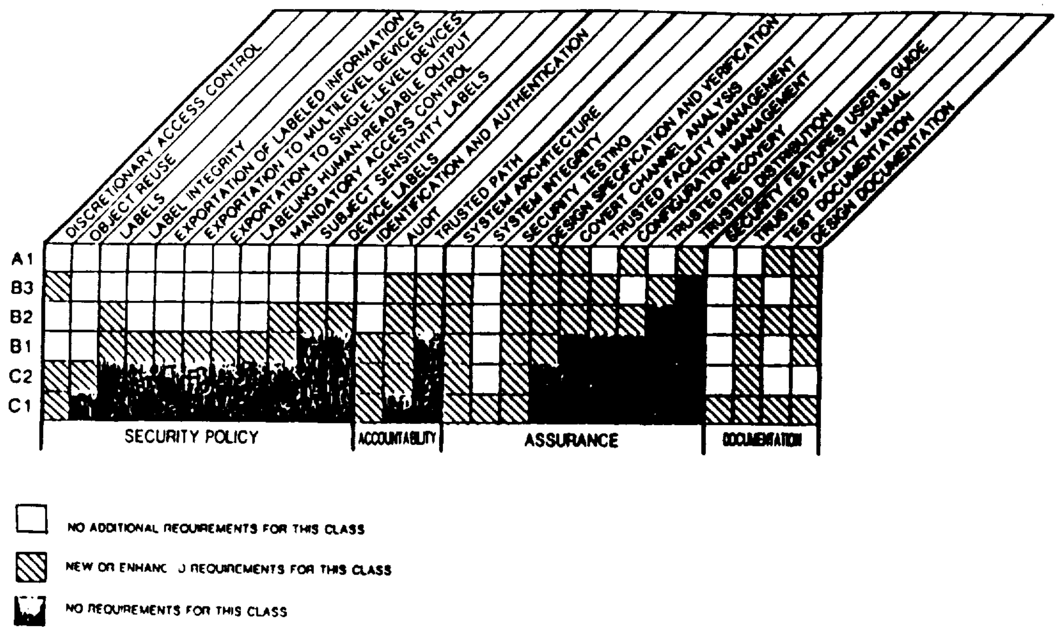
Multilevel Security On Networks

Bell-LaPadula Security Properties

Multilevel Secure Network must preserve the following two properties of access to data:

- Simple Security Property - no user may read data at a level higher than that for which the person is authorized
- *-Property - no person may write data to a level lower than what the person has accessed

TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA
SUMMARY CHART



taken from the "Orange Book," <u>DoD Trusted Computer System Evaluation Criteria</u> for use as a viewgraph ONLY

Network Security Overview

- New Issues
- Familiar Solutions

  Encryption

  Access Controls

  Authentication

  (many-same as operating systems)

(Schneider and Pfleeger)

Advantages of Computing Networks

Advantages over Single Processor Systems

1. **Resource sharing.**

   Users of a network can access a variety of resources through the network.

   Sharing may justify existence of resources which are costly and not frequently used.

   Reduced maintenance and storage costs.

2. **Increased reliability.**

   Redundancy of computing systems.

3. **Distributing the workload.**

   Workload can be shifted from a heavily loaded system to an underutilized system.

4. **Expandability.**

   Network systems can be expanded easily by adding new nodes.

Network Security Issues

<u>Reasons for Network Security Problems</u>

1. **Sharing.**

   More users have access to network systems.

   Access is afforded to many computing systems.

2. **Complexity of System.**

   Network operating/control system is very likely to be more complex than an operating system for a single computing system.

   A network may combine two or more possibly dissimilar operating systems with mechanisms for interhost connection.

3. **Unknown perimeter.** **(See Figure 10.8, page 374)** Pfleeger

   Host may be a node on multiple networks.

   Unknown or uncontrolled group of possibly malicious users.

   New hosts may be added to the network at any time.

   User on a host is unaware of the potential connections from users of other networks.

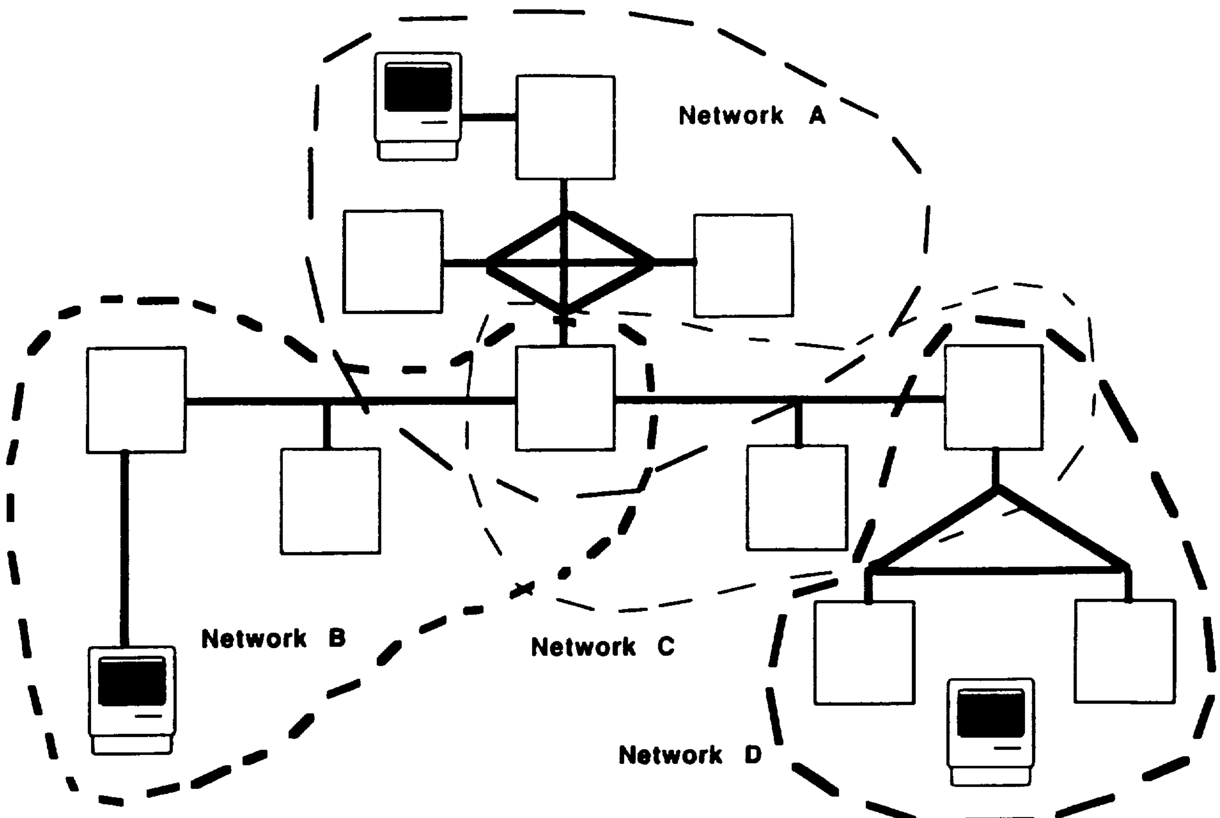


Figure 10.8 Unclear Network Boundaries

12/10/88 Pfleeger

Network Security Issues

<u>Reasons for Network Security Problems</u>

4. **Many points of attack.**

    File may pass through many host machines to get to user.

    Administrator of one host has no control over other hosts in the network.

    User has to trust the access control mechanisms of all systems within network.

5. **Unknown path.** **(See Figure 10.9, page 375)** Pfleeger

    May be many paths from one host to another.

    Network users seldom have control over the routing of messages.

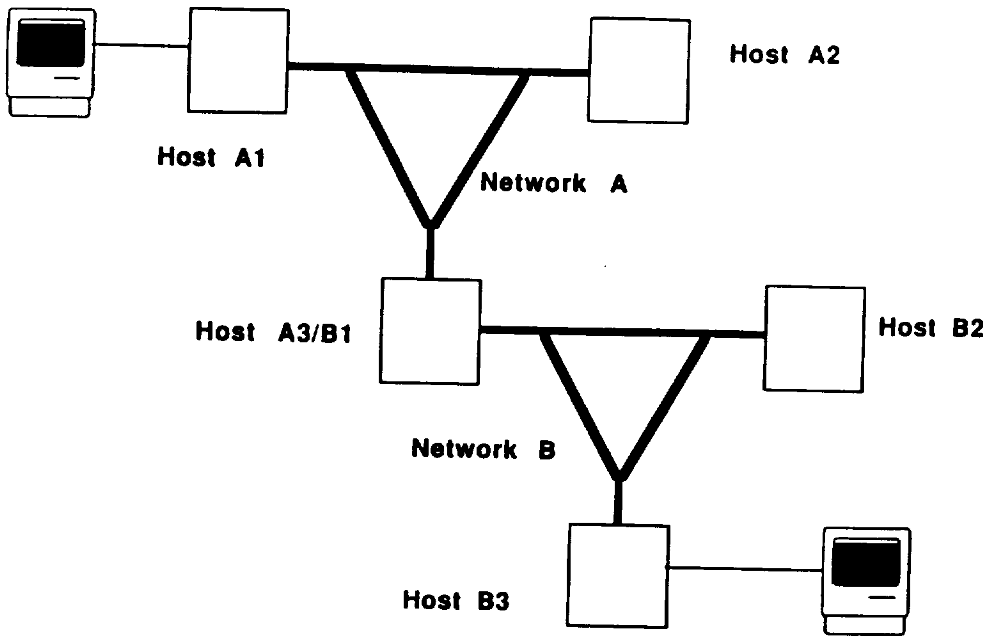


Figure 10.9 Message Routing in a Network

12/10/88 (Pfleeger, Security in Computing)

Security Exposures

1. **Privacy.**

   With many unknown users on a network, concealing sensitive data becomes difficult.

2. **Data Integrity.**

   Because of many nodes and many users, the risk of data corruption is higher.

   Corruption includes:

   - modification of messages
   - insertion of bogus messages
   - deletion of messages
   - replay of messages
   - reordering of messages

3. **Authenticity.**

   Difficult to assure the identity of a user on a remote system.

   Network may not be able to trust the authenticity of the hosts themselves.

4. **Covert Channels.**

   Networks offer possibilities for construction of covert channels for data flow.

Encryption in Networks

- **Link Encryption**

  - data is encrypted just before it is placed on the physical communications link.

  - encryption occurs at layer 1 or 2 in the OSI model.

  - decryption occurs just as the communications enters the receiving computer.

  - especially vulernable when a communication must pass through one or more additional hosts between sender and receiver.

  - appropriate when the transmission line is the point of greatest vulernability.

Encryption in Networks

- **End-to-End Encryption**

  - provides security from one end of a transmission through the other.

  - can be applied by a hardware device between the user and the host.

  - can be done by software running on the host computer.

  - encryption is performed at the highest levels - either layer 7 (application) or layer 6 (presentation) of the OSI model.

  - messages sent through several hosts are protected.

Comparison of Encryption Methods

- **Link Encryption**
  - Cryptographic facility is involked for all transmissions along a particular link.
  - every host receiving communications must have a cryptographic facility to decrypt messages.
  - all hosts must share keys
  - authenticates only the node, not the user.
  - faster, easier for the user, and uses fewer keys.

- **End-to-End Encryption**
  - is applied to "logical links", which are channels between two processes.
  - no need for cryptographic facilities.
  - encryption is used only for those messages and applications which need it.
  - encryption can be done with software.
  - numerous keys may be required to provide adequate security between multiple users.
  - more flexible, can be used selectively, and can be customized to the application.

Access Control in Networks (Pfleeger)

- Port Protection
  - dial-in port access is a serious vulnerability to a network
- Automatic Call-Back
  - computer breaks the communication connection and calls the user back after consulting an internal table of telephone numbers
  - works for users who expect to be at one phone number/location
- Differentiated Access Rights
  - limit the locations from which access is allowed
  - access to sensitive items/data only by direct connection, not through another network host
- Silent Modem
  - computer remains silent until calling system initiates connection sequence
- Node Authentication

User Authentication

Authentication mechanisms are divided into three categories :

1. What you know, such as a password or encryption key.

2. What you possess, such as a token or a capability.

3. Somthing about you, such as a picture or a fingerprint.

Passwords

- Composed of letters, digits, and other characters
- Long (many possibilities, requires an exhaustive attack)
- Not a common word or name (to foil the dictionary attack)
- Unlikely choice of words, not an address or family name
- Passphrases
- Challenge-Response Systems
- Frequently changed
- Not written down

INTRUSION DETECTION SYSTEMS

- After authentication, monitor user with an I/D system.
- Ultimately you want automatic online real–time monitoring and early warnings
- Difficult: some users have erratic work hours, habits, locales

EARLY WORK

Automating Offline Log Analysis

- SIDS (SRI): Building automated tools for audit trail security analysis (1986)
- Sytek: Building special security audit trails (1986)

Sytek Work

- Tried to show feasible a tool that ranked user sessions by suspiciousness
- For each *user*, expected values for *features* were determined by *training*
- If feature outside user's range or set of expected values, suspicious
- Assumes user *profiles*

DISCRIMINATING FEATURES

as found by Sytek

- Password changed
- User identity queried
- Access to system dictionary
- Device on which accessed file resides
- File size
- Oversized file associated with this command
- Group ID of owner of accessed file
- User ID of owner of accessed file
- Time of use
- Day of use
- User program CPU time
- Maximum program memory use

(all under 15% false alarms)

I/D SYSTEM SURVEY

Intrusion Detection Expert System

- IDES – SRI
- Model independent of any particular target system, application environment,
- ... system vulnerability, or type of intrusion.
- Prototype runs on a Sun and monitors login, logout, program execution,...
- directory modification, file access, system call, session location change,...
- and network activity.
- Two types of measures: discrete (e.g., time of login) and ...
- continuous (e.g., connect time (count accumulates over a user session))
- User behavoir profiles for each measure
- Profile data aged with half-life of 50 days (gives window of behavoir for user)
- As users change behavoir, thresholds in profiles change
- Currently monitors a DEC-2065 running TOPS-20 operating system
- IDES has flexible system-independent audit record format
- As of Fall '88, flagged 60,000 items; found 5 hits (illegal resource use)
- Goal: detect intrusion in 5 sec. for 100 user load
- IDES-88: 36 measures now, 25 on users, 5 target system, 6 hosts
- Allows groupings of users, remote hosts, times, days, target systems

MIDAS -NCSC

- Being developed to monitor Multics
- Uses home-grown expert system shell, forward-chaining inference engine, and
- ... an explanation facility.
- Profiles maintained in Lisp.
- Lisp rules are compiled for speed.
- MIDAS has 40 rules (1988)
- Four types of heuristic rules:
- - immediate (use no info, these events in isolation are suspicious)
- - anomaly (w.r.t. user or remote system behavoir)
- - system-wide state (unusual system activity)
- - sensitive path (NOT YET) Is a user command sequence likely to be an attack?

MIDAS Rules

- Try to detect break-ins
- - password failure on system account
- - login failure with unknown user name
- - login attempt from outside continental U. S.
- - login attempt to a locked account
- Try to detect masqueraders
- - unusual time, location, terminal type for this user
- - invalid commands
- - logged in simultaneously from different locations
- Also checks for resource overuse, inactive session, mods to sys files

DISCOVERY (TRW)

- What if users give away their passwords, etc.?
- DISCOVERY attempts to detect imposters
- Only analyzes correct inquiries submitted by customers, not errors
- Not real-ime
- consumer credit queries with established pattern
- 150,000 users; 450,000 to 1 millino queries daily
- 60 days min. to develop adequate user profile
- 50-100 followups generated daily - 3 violations found in 1-2 yrs.
- Developed on Sun, ported to AT, currently runs on 3094 in Cobol

Clyde Digital Systems Audit

- Audits VAX/VMS Users
- Saves every byte in a file that passes between terminal and system
- Audit can be done for specific users, times, or programs
- Three reports: a summary security report related to high-risk users
- A security event report
- Supporting data for the first two reports
- Risk factors: unusual hours, sessions with AUTHORIZE or SYSGEN utilities;...
- ... browsing; file access alarms; repeated unsuccessful logins; sessions
- ... with dial-up or remote terminals; simultaneous logins for same user; ...
- attempts to turn off logging. Weights used for each one.

KEYSTROKE DYNAMICS

- Based on "fist of the sender" concept from Morse Code days
- Based on typing characteristics read by a board in CPU socket of IBM PC mother
- ... board. No special keyboard is needed.
- Registration: user types password 12 times
- Testing: can even be done continuously by BioContinuous product
- Characteristics used inlcude intervals, rhythm, a pressure analog, and error
- ... characteristics. These form an electronic signature.
- User's keystroke dynamics are then continuously and automatically checked.
- Future: all electronic signatures will be stored in one place centrally on net

Wisdom and Sense – LANL

BIOMETRICS

- Identification on basis of physical characteristics
- Access security biometric:
-     – personal or physical charcteristic
-         – measured accurately
- Device must "know" users: read, store, retrieve, and compare
- Quick and reliable
- Digital representation

Examples of Biometric Checks

- Fingerprint: best stored image is not photo, swirls and ridges
- Signature: Letter spacings, loops, angles, speed, and pressure
- Voice: Fourier transform, known phrase

BIOMETRICS

- Chosen because characteristics do not change with time

OLD SOLUTIONS OFTEN THE BEST!

Don't only do technical fixes!

- Policy and procedures
- Backups
- Training
- Auditing

FOR FURTHER ASSISTANCE . . .

- Government: OPM course and documents, NIST, NCSC, CERT, Balt/Wash conference
- Private training courses: CSI, MIS, ACM
- Three-day short courses offered by universities. Also their regular courses
- Consultants (be sure to get three references and beware low-ball bids)...
- ... You get what you pay for.

REFERENCES

Baskerville, Richard, Designing Information Systems Security, Wiley, 1988.

Pfleeger, Charles P., Security in Computing, Prentice-Hall, 1989.

Proc. 1989 Computer Security Risk Management Model Builders Workshop, available from National Institute of Standards and Technology

Guideline for Automatic Data Processing Risk Analysis, FIPS PUB 65, National Bureau of Standards, 1979.

Raiffa, Howard, Decision Analysis, Addison-Wesley, 1968.

Hoffman, Lance J., "Smoking Out the Bad Actors: Risk Analysis in the Age of the Microcomputer", Computers and Security 8 (1989), pp. 299-302.

Schmucker, Kurt J., Fuzzy Sets, Natural Language Computations, and Risk Analysis, Computer Science Press, 1984.

Ellis, A. and Garrabrants, M., CERTS (1990 M.S. thesis, Naval Postgraduate School, Monterey, CA) (a methodology for evaluating automated and non-automated risk analysis tools)

Lunt, Teresa, "Automated Audit Trail Analysis and Intrusion Detection: a Survey", Proc. 11th Nat. Comp. Security Conf., 1988, pp. 65-73.

Lunch Speaker

Computers and the Law

Michael Gemignani

University of Houston-Clear Lake

Notes

Notes

PRECEDING PAGE BLANK NOT FILMED

Evolution - User Computing Security

Emily Lonsford

MITRE

EVOLUTION OF USER COMPUTING AND SECURITY

Emily H. Lonsford

EMILY H. LONSFORD is a member of the Technical Staff at The MITRE Corporation. She leads the MITRE team which supports the AIS Security Engineering Office at NASA's Johnson Space Center.

Notes

Notes

Managerial

Security in Software Applications and Development

James Molini

Computer Sciences Corporation

117 - 118

James E. Molini
Computer Sciences Corp.
Houston, TX

Mr. Molini has been providing technical computer security support to NASA for over 5 years. In is current position he performs the Independent Security Self Assessment for the Space Shuttle Program's Software Development Facility. Previous to that, he provided technical support to the Computer Security Officer at the NASA Lewis Research Center in Cleveland, OH. In that position, he supported the design and implementation of the Center AIS program.

He began working in computer security in the U.S. Army at Ft. Bragg, NC. Since that time he has supported a variety of classified and unclassified computer security initiatives for a number of organizations.

INTEGRATING SECURITY INTO SOFTWARE DEVELOPMENT AND SOFTWARE QUALITY ASSURANCE

ISIS SYMPOSIUM
MAY 16, 1990
HOUSTON, TEXAS

PRESENTED BY
JAMES E. MOLINI
COMPUTER SCIENCES CORP.

THE BENEFITS OF SOFTWARE

HARDWARE DEGRADES OVER TIME.
PEOPLE DEGRADE OVER TIME.
SOFTWARE DOES NOT.

Software is either logically correct, or it isn't.
This can be tested during development.

IW-IECJ

OVERVIEW

- SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)
- SECURITY ACTIVITIES IN SDLC
- SOFTWARE QUALITY ASSURANCE (SQA)
- SECURITY ACTIVITIES IN SQA

ISSASW1

OVERVIEW

PURPOSE OF SDLC

- DIVIDES SOFTWARE DEVELOPMENT PROCESS INTO PHASES
- SDLC OBJECTIVES
  - STRUCTURED MANAGEMENT SCHEME FOR CONTROLLING COSTS & SCHEDULES
  - ASSURE PROPER AND RESPONSIVE COMMUNICATION CHANNELS AMONG USERS, DESIGNERS, DEVELOPERS, TESTERS, ... AND

INFORMATION SYSTEM SECURITY MANAGERS

ISSASW2

SDLC PHASES

- PREDEFINITION PHASE
- PROJECT DEFINITION PHASE
- SYSTEM DEVELOPMENT PHASE
- IMPLEMENTATION PHASE

ISSASW3

SECURITY ACTIVITIES
RESPONSIBILITIES

ACTIVITIES	ACCOMPLISHED BY
Security Devel Plan	PROJ MGR/ISS
Data/Appl Sensitivity	USERS/ISS
Sec Risk Assessment	USERS/ISS
Security Objectives	USERS/ISS System Planners
Security Feasibility	USERS/ISS System Planners
Security Requirements	USERS/ISS System Planners

ISSASW4

ACTIVITIES	ACCOMPLISHED BY
SECURITY TEST PLAN	QA, V&V, ISS SYSTEM DEVELOPERS
SECURITY SPECIFICATIONS	SYSTEM DEVELOPERS
SECURITY TEST PROCs	QA, V&V, ISS
SECURITY RELATED CODE	PROGRAMMERS SYSTEM DEVELOPERS
DOCUMENT SAFEGUARDS	PROGRAMMERS SYSTEM DEVELOPERS
SECURITY TEST & EVAL	QA, V&V, ISS
SECURITY TEST REPORT	QA, V&V, ISS
CERTIFICATION STATEMENT	ISS

SECURITY ACTIVITIES
DATA/APPL SENSITIVITY

- NATURE OF DATA
- TYPES OF FUNCTIONS TO BE PERFORMED
- FOR EXAMPLE;
  - PERSONAL DATA
  - HIGH DOLLAR VALUE ASSETS
  - CRITICAL FORMULAS
  - ALGORITHMS
  - ENGINEERING CALCULATIONS

ISSASW8

DATA/APPL SENSITIVITY
SENSITIVITY LEVELS -EXAMPLE

- LEVEL 0 - Negligible effect on NASA missions or functions due to inaccuracy, alteration, disclosure or unavailability
- LEVEL 1 - Minimal impact on agency functions; damage to agency's image or reputation; loss of tangible asset or resource
- LEVEL 2 - Adversely affect the conduct of a NASA program; significant damage to agency's ability to fulfill a statutory responsibility; or impact between $100,000 and $10,000,000
- LEVEL 3 - Pose a threat to human life; irreparable damage to NASA's ability to carry out an essential mission or function; or impact of more than $10,000,000

ISSASW9

SECURITY ACTIVITIES
ASSESS RISKS*

- IMPACT OF FAILURES
  - INACCURATE DATA
  - FALSIFIED DATA
  - DISCLOSED DATA
  - INACCURATE PROCESSES, FUNCTIONS
  - LOST DATA, APPLICATION CODE OR DOCUMENTATION
  - UNAVAILABILITY OF DATA OR APPLICATION

* For each proposed alternative

ISSASW11

SECURITY ACTIVITIES
SECURITY OBJECTIVES

- DATA INTEGRITY
- APPLICATION INTEGRITY
- DATA CONFIDENTIALITY
- APPLICATION CONFIDENTIALITY
- RESOURCE AVAILABILITY

ISSASW12

SECURITY ACTIVITIES
SECURITY FEASIBILITY

- BASED ON SECURITY OBJECTIVES
- ARE SAFEGUARDS AVAILABLE?
- HOW WELL WILL THEY SATISFY THE SECURITY OBJECTIVES?
- SHOULD SAFEGUARDS BE:
  - PREVENTATIVE ?
  - DETECTIVE?
  - RECUPERATIVE?
- WHAT MIX OF ADMINISTRATIVE, PHYSICAL OR TECHNICAL SAFEGUARDS IS APPROPRIATE?

ISSASW13

SECURITY CONTROLS
SOFTWARE vs. PROCEDURAL CONTROLS

- Software controls are harder to design, but easier to implement than procedural controls.
- Software controls can be tested earlier and more frequently during development than procedural controls.
- Procedural controls usually become less effective over time.

ISSW-A1

SECURITY ACTIVITIES
SECURITY REQUIREMENTS

- IDENTIFICATION AND DEFINITION OF SYSTEM INTERFACES
- IDENTIFICATION OF SENSITIVE OBJECTS
- DETERMINATION OF ERROR TOLERANCES
- AVAILABILITY REQUIREMENTS

Guildelines for Security of
Computer Applications
NIST FIPS PUB 73

ISSASW16

SECURITY REQUIREMENTS
BASIC CONTROLS*

- DATA VALIDATION
- USER IDENTITY VERIFICATION
- AUTHORIZATION
- JOURNALING
- VARIANCE DETECTION
- ENCRYPTION

*Guidelines for Security of
Computer Applications
FIPS PUB 73

ISSASW17

APPLICATION SECURITY
SELECTION OF CONTROLS

- Security should be provided by the environment (eg. OS, DBMS).
- Environmental controls are usually better defined and more comprehensive.
- Applications should request security through approved interfaces (eg. system calls, macros).
- Using this technique, security interfaces can be defined during application design.

APPLICATION SECURITY
APPLICATION SPECIFIC SECURITY TOOLS

- Application security tools should be designed to supplement, not replace environmental security controls.
- If environmental security controls are defective, fix, or replace the environment.
- IF YOU DEVELOP SECURITY CODE, YOU MUST MAINTAIN IT.

SECURITY TEST PLAN
OVERVIEW

- FROM A SECURITY PERSPECTIVE TESTING OF SECURITY CONTROLS SHOULD FOCUS ON ENSURING THAT SECURITY CONTROLS ARE:
  - INVOKED WHEN REQUIRED
  - CANNOT BE EASILY BYPASSED
  - AUDITABLE
  - APPROPRIATE IN VIEW OF THE SENSITIVITY OF THE DATA OR THE APPLICATION

ISSASW18

SECURITY TEST PLAN

- CONTENTS
  - WHAT IS TO BE TESTED
  - TESTING SCHEDULE
  - RESOURCE REQUIREMENTS
  - TESTING MATERIALS
  - REQUIREMENTS FOR TEST TRAINING
  - TEST LOCATION
  - TESTS TO BE PERFORMED AND THE RELATIONSHIP TO THE FUNCTIONAL SECURITY REQUIREMENTS
  - TESTING METHODOLOGY
  - EVALUATION CRITERIA
  - DATA REDUCTION TECHNIQUES
  - DESCRIPTION OF EACH TEST TO BE PERFORMED

ISSASW19

SECURITY TEST PLAN
TYPES OF EVALUATION

- NIST FIPS PUB 102 - "Guidelines for Computer Security Certification and Accreditation"
- BASIC EVALUATION: Concerned with the overall functional security posture.
  - Verify that security functions exist
  - Implementation method is of sufficient quality to be relied upon
- DETAILED EVALUATION: Concerned with whether security functions work properly.
  - Satisfy performance criteria
  - Acceptably resist penetration

ISSASW22

SECURITY ACTIVITIES
DOCUMENTATION

- DOCUMENTATION IS THE PROCESS OF DESCRIBING WHAT FUNCTIONS AN APPLICATION PERFORMS, HOW IT PERFORMS THEM, AND HOW THE FUNCTIONS ARE TO BE USED.
  IN OTHER WORDS:

  "A CLEAR MEANS OF UNDERSTANDING ALL ASPECTS OF THE APPLICATION SYSTEM"

  TO INCLUDE SECURITY CONTROLS!

ISSASW24

SECURITY & SQA
OVERVIEW

- ONE OF THE AREAS WHERE SOFTWARE ERRORS CAN BE LEAST TOLERATED IS THAT OF SECURITY SAFEGUARDS.
- ONE OF THE TECHNIQUES BEING EMPLOYED TO IMPROVE THE RELIABILIITY OF SOFTWARE:

SOFTWARE QUALITY ASSURANCE!

- SQA CAN BE EMPLOYED TO REDUCE THE POTENTIAL FOR INCORPORATING UNRELIABLE SECURITY SAFEGUARDS IN APPLICATIONS.

ISSASW26

SOFTWARE QUALITY
FACTORS

- CORRECTNESS - The extent to which a safeguard satisfies its specification & fulfills the application security objectives.
- RELIABILITY - The extent to which a safeguard can be expected to perform its intended function with required precision.
- EFFICIENCY - The amount of computing resources and code required by a safeguard to perform its function.
- INTEGRITY - The extent to which access to the safeguard by authorized persons can be controlled.
- USABILITY - The effort required to learn, operate, prepare input or interpret output from a safeguard.

ISSASW28

SOFTWARE QUALITY
FACTORS

- MAINTAINABILITY - The effort required to locate and fix an error in, or to determine the impact of other system changes, on a safeguard.
- TESTABILITY - The effort required to test or audit a safeguard to ensure that it performs its intended function.
- FLEXIBILITY - The effort required to modify an operational safeguard.
- INTEROPERABILITY - The effort required to couple or integrate safeguards into the application system.

ISSASW29

SECURITY SPECIFICATIONS
OVERVIEW

PRIMARY SOURCE OF SECURITY PROBLEMS IS:

COMPLEX DESIGN

THAT CANNOT BE IMPLEMENTED

CORRECTLY OR EASILY,

NOR MAINTAINED OR AUDITED!

KEEP IT SIMPLE!!!

ISSASW20

SECURITY SPECIFICATIONS
OVERVIEW

- SUGGESTIONS FOR DESIGN OF SECURITY CONTROLS:
  - No Unnecessary Programming
  - Restricted User Interfaces
  - Human Engineering
  - No Shared Computer Facilities
  - Isolation of Critical Code
  - Backup and Recovery
  - Use of Available Controls

ISSASW21

APPLICATION SECURITY
GENERAL RULES OF THUMB

- When defining security requirements, a picture is worth 10,000 words. If it can't be described with a picture, it can't be implemented.
- Product testing never takes less than 2 months. Make sure that adequate test time is allocated for security controls and interfaces.
- Compressing a delivery schedule always increases errors. It is better to deliver less security functionality than to produce error laden security code.
- Undocumented security code can be worse than no code at all.
- Security is not more important than correct operation of the system

ISSW-A4

Managerial

Risk Management

F. G. Tompkins

Computer Sciences Corporation

PRECEDING PAGE BLANK NOT FILMED

BIOGRAPHICAL SKETCH

FREDERICK G. TOMPKINS
COMPUTER SCIENCES CORPORATION
HOUSTON, TEXAS

Frederick G. Tompkins is the Manager of the AIS Security Program Office (SPO) responsible for the overall operation and administration of the automated information systems security functions for the MOSC (Mission Support Directorate and Operations Support Contract) at Johnson Space Center. His primary duties include establishing policies, standards and procedures for assuring the security and integrity of NASA sensitive systems and data that are processed on MOSC managed and operated ADP systems.

Mr. Tompkins has over 30 years of experience in the fields of security, intelligence, data processing and automated information security. He has served as consultant to a number of Federal and State government agencies and to a number of private businesses. He has authored a number of guidelines and methodological approaches for risk management, contingency planning, security in the software development life cycle, and training.

Mr. Tompkins has been a member of ASIS since 1977 and has chaired the ASIS Standing Committee on Computer Security and has served as member of the Society's Standing Committees on Privacy and Information Management, Disaster Management and Safeguarding Proprietary Information. He has served as an Advisor to the DataPro Reports on Information Security since its initial publication. Mr. Tompkins is a member of the Computer Security Institute.

Mr. Tompkins holds an Associate Arts degree from Orange Coast College and a Bachelor of Science in Technology of Management from The American University.

Current Address:

Frederick G. Tompkins
Manager, MOSC AIS Security Program Office
Computer Sciences Corporation
16511 Space Center Boulevard
MS R20
Houston, TX 77058
Phone: 713-282-2203
FAX: 713-282-2266

INFORMATION SECURITY AND INTEGRITY SYSTEMS SYMPOSIUM

RISK MANAGEMENT

FREDERICK G. TOMPKINS

MAY 15-16, 1990
HOUSTON, TEXAS

RISK MANAGEMENT

- RISK MANAGEMENT APPROACH
- RISK ANALYSIS
- RISK REDUCTION ANALYSIS
- MANAGEMENT DECISION
- RISK REDUCTION ACTIONS
- IMPLEMENTATION & MAINTENANCE
- RISK MANAGEMENT PLANS
- RISK MANAGEMENT TOOLS

ISIS-1

RISK MANAGEMENT

- FUNDAMENTAL PREMISE:
  "IT IS NOT POSSIBLE TO
  HAVE A RISK-FREE
  DATA PROCESSING
  ENVIRONMENT:
  THEREFORE:
  RISKS MUST BE MANAGED!

ISIS2

RISK MANAGEMENT

- RISKS MUST BE:
  - APPROPRIATELY DEFINED
  - CATEGORIZED AS TO LIKELIHOOD OF OCCURRENCE
  - ASSESSED AS TO RESULTANT CONSEQUENCES/IMPACTS
- RESOURCES MUST BE ALLOCATED TO MINIMIZE RISKS
- RISK MANAGEMENT IS A SPECIALIZED APPLICATION OF THE SYSTEMS APPROACH TO PROBLEM SOLVING

ISIS3

SYSTEMS APPROACH

- THE SYSTEMS APPROACH CONCENTRATES ON ANY SYSTEM AS A WHOLE RATHER THAN THE PARTS AND RELATES THE PARTS TO EACH OTHER TO ACHIEVE THE TOTAL SYSTEM GOALS

ISIS4

RISK MANAGEMENT
PROCESS

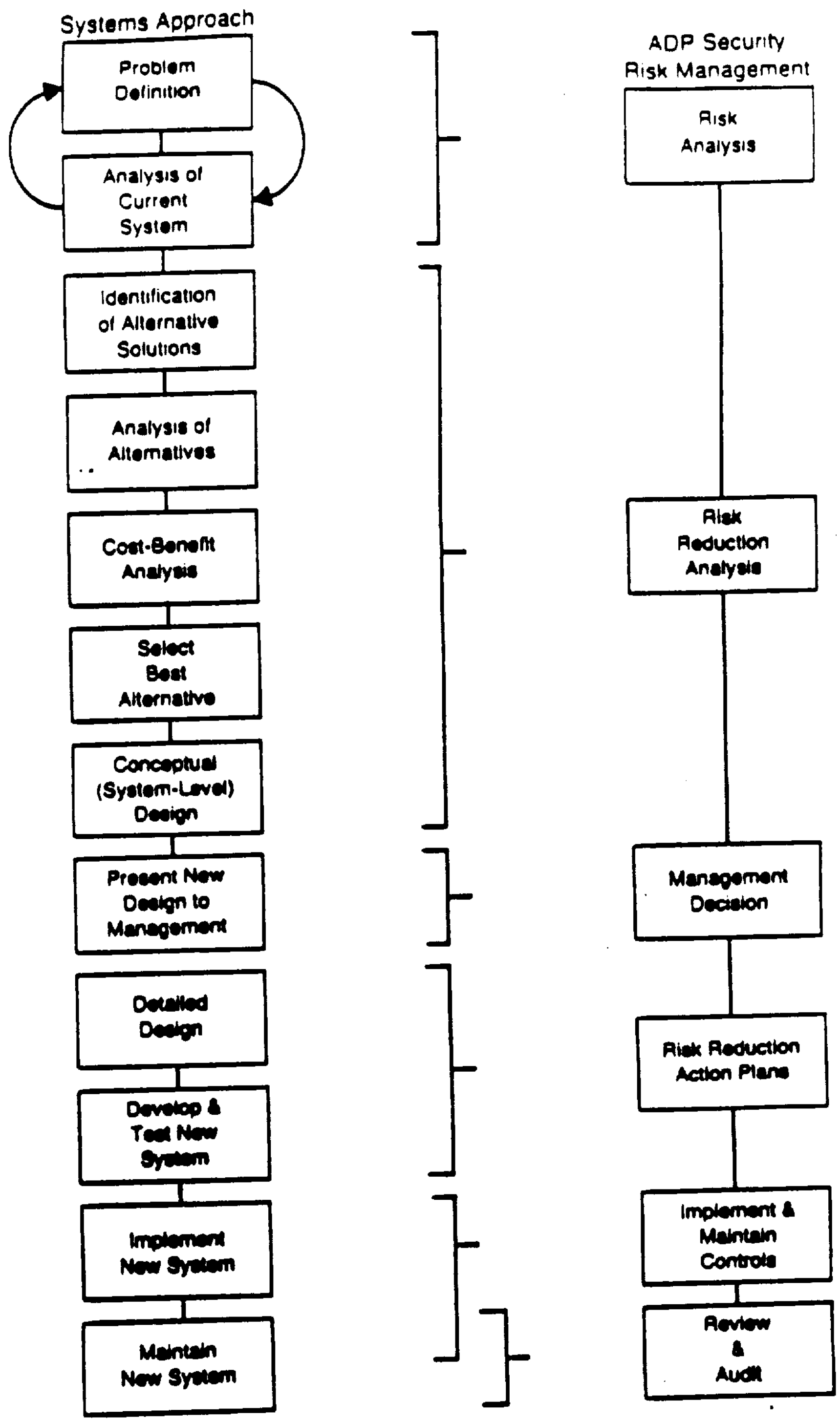
- AIS SECURITY RISK MANAGEMENT PROCESS IS DESIGNED TO ANSWER THE FOLLOWING:
  - WHAT IS AT RISK?
  - WHAT ARE THE IMPACTS?
  - WHAT CONTROLS ARE AVAILABLE TO REDUCE RISKS?
  - WHAT CONTROLS WILL PROVIDE THE BEST RETURN ON INVESTMENT?
  - WHO IS RESPONSIBLE FOR IMPLEMENTATION?
  - HOW WILL CONTROLS BE IMPLEMENTED?
  - HOW EFFECT ARE CONTROLS OVER TIME?

ISIS5

RISK MANAGEMENT
PHASES

- RISK ANALYSIS
- RISK REDUCTION ANALYSIS
- MANAGEMENT DECISION
- RISK REDUCTION ACTION PLANS
- IMPLEMENTATION/MAINTENANCE
- REVIEW AND AUDIT

ISIS6



THE ADP SECURITY RISK MANAGEMENT PROCESS IN RELATIONSHIP TO THE SYSTEMS APPROACH

RISK ANALYSIS
DATA COLLECTION & ANALYSIS

- SCOPING - PARTS OF THE FACILITY & TYPES OF ASSETS TO BE INCLUDED
- EXPOSURE ZONES PHYSICAL VS LOGICAL
- VALUE OF ASSETS - REPLACEMENT $
- THREATS TO RISK ENVIRONMENT (PHYSICAL, THEFT, ACTS OF NATURE)
- ANNUAL FREQUENCY OF OCCURRENCE FOR EACH THREAT
- VULNERNERABILITIES (WALKTHROUGHS. INTERVIEWS, INSPECTIONS, CHECKLISTS)
- DOCUMENT CONTROLS IN PLACE & EFFECTIVENESS (IF POSSIBLE)

ISIS8

RISK ANALYSIS

IMPACTS/CONSEQUENCES

- QUALITATIVE STATEMENTS OF LOSS
- QUANTITIATIVE STATEMENTS OF LOSS - ANNUAL LOSS EXPOSURE ($)

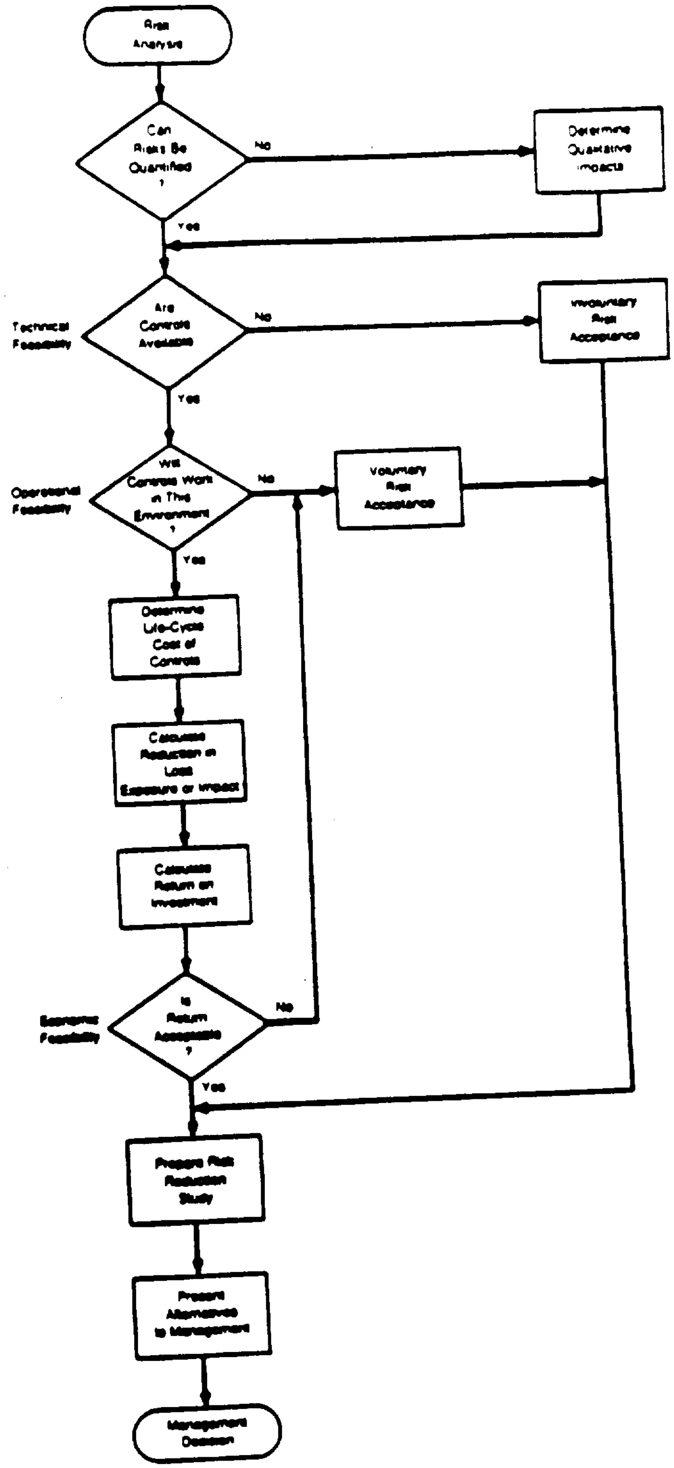
$$\frac{\text{ONE-TIME LOSS(\$)}}{\text{THREAT OCCURRENCE RATE}}$$

ISIS9

RISK REDUCTION ANALYSIS

- OBJECTIVE: DETERMINE ACCEPTABILITY/UNACCEPTABILITY OF IDENTIFIED RISKS
- VOLUNTARY VS INVOLUNTARY RISK ACCEPTANCE
- AVAILABILITY OF SAFEGUARDS (TECHNICAL FEASIBILITY)
- OPERATIONAL FEASIBILITY
- ECONOMIC FEASIBILITY (COST/BENEFIT - ROI)
- RISK REDUCTION DECISION STUDY FOR MANAGEMENT

ISIS10


Risk Analysis
Can Risks Be Quantified ?
No
Determine Qualitative Impacts
Yes
Technical Feasibility
Are Controls Available
No
Involuntary Risk Acceptance
Yes
Operational Feasibility
Will Controls Work in The Environment ?
No
Voluntary Risk Acceptance
Yes
Determine Life-Cycle Cost of Controls
Calculate Reduction in Loss Exposure or Impact
Calculate Return on Investment
Economic Feasibility
Is Return Acceptable ?
No
Yes
Prepare Risk Reduction Study
Present Alternatives to Management
Management Decision

RISK REDUCTION

- AVAILABILITY OF CONTROLS (TECHNICAL FEASIBILITY)
- CONTROL OBJECTIVES
  - PREVENTION CONTROLS
  - DETECTION CONTROLS
  - RECOVERY CONTROLS
- TYPES OF CONTROLS
  - ADMINISTRATIVE
  - PHYSICAL
  - TECHNICAL

ISIS11

TECHNICAL FEASIBILITY

	PREVENT	DETECT	RECOVER
ADMIN			
PHYSCIAL			
TECHNICAL			

ISIS12

OPERATIONAL FEASIBILITY

- IMPACT ON CURRENT OPERATIONS
- HOW WILL PERSONNEL RESPOND TO THE CHANGES BROUGHT ON BY IMPLEMENTING THE CONTROL
- TRAINING REQUIREMENTS?

ISIS13

ECONOMIC FEASIBILITY

- DETERMINE THE:
  - INVESTMENT COSTS
  - IMPLEMENTATION COSTS
  - OPERATING OR MAINTENANCE COSTS
  - CONSIDER BOTH DIRECT & INDIRECT COSTS

- COST-BENEFIT (OR ROI) WHERE POSSIBLE - INTANGIBLE BENEFITS (BETTER MANAGEMENT CONTROL)

$$\text{ROI} = \frac{\text{ORIGNINAL ALE} - \text{REVISED ALE}}{\text{ANNUAL COST OF NEW CONTROL}}$$

ISIS14

RISK REDUCTION
DECISION STUDY

- OBJECTIVE: OBTAIN FAVORABLE DECISION FROM MANAGEMENT
- RECOMMENDATIONS - ACCEPTABILITY
- ALTERNATIVES - CONTROLS

ISIS15

RISK REDUCTION
DECISION STUDY

- REPORT CONTENTS:
  - Executive Summary
  - Summary of Risk Scenarios
  - Technical Feasibility Analysis
  - Operational Feasibility Analysis
  - Economic Feasibility Analysis
  - Acceptable/Unacceptable Risks
  - Control Alternatives

ISIS16

MANAGEMENT DECISION

- FACILITY MANAGEMENT DETERMINES WHICH RISKS ARE ACCEPTABLE OR UNACCEPTABLE
- EVALUATES ROI's
- DETERMINES WHICH OR THE ALTERNATIVES WILL BE IMPLEMENTED
- DECISIONS SHOULD BE DOCUMENTED

ISIS20

MANAGEMENT DECISION

- MANAGEMENT OPTIONS:
  - ELIMINATE RISK
  - LOSS PREVENTION
  - LOSS LIMITATION
  - LOSS TRANSFER
  - ACCEPT THE RISK

ISIS17

ACTION PLANS

- ADMINISTRATIVE CONTROLS -
  - WRITTEN, STAFFED, PUBLISHED & DISSEMINATED
- PHYSICAL CONTROLS -
  - PROCURED, TESTED, IMPLEMENTED; TRAINING
- TECHNICAL CONTROLS -
  - DESIGNED & DEVELOPED OR PROCURED, TESTED & IMPLEMENTED; TRAINING
- RISK REDUCTION ACTION PLANS SHOULD BE USED TO PROVIDE MANAGEMENT/PROJECT CONTROL (E.G., GANTT CHARTS)

ISIS18

IMPLEMENTATION & MAINTENANCE

- MAY REQUIRE CHANGES IN PROCESSES, FUNCTIONS & RESPONSIBILITIES
- TO BE EFFECTIVE, CONTROLS WILL REQUIRE CONSTANT & CONSISTENT USE
- CHANGES IN OPERATIONAL PROCEDURES MUST BE COORDINATED
- MAINTENANCE PROCEDURES WILL BE REQUIRED TO RESPOND TO CHANGES WHICH MAY MANDATE MODIFICATION TO CONTROLS

ISIS19

REVIEW & AUDIT

- PERIODIC EVALUATION OF EFFECTIVENESS DUE TO CHANGES IN ENVIRONMENT, TYPES OF APPLICATIONS, OR MANAGEMENT FOCUS
- NEW TECHNOLOGY - INCLUDING SECURITY TECHNOLOGY
- CHANGES MAY DICTATE NEED TO UPDATE RISK AND/OR RISK REDUCTION ANALYSES

ISIS20

RISK MANAGEMENT PLANS

- RISK MANAGEMENT IS AN ONGOING PROCESS
- RISK MANAGEMENT PLANS IS A RISK REDUCTION ACTION ITEM

ISIS21

RISK MANAGEMENT PLANS

- RISK MANAGEMENT PLAN SHOULD INCLUDE:
  - DESCRIPTION OF RISK ENVIRONMENT
  - THREATS
  - THREAT OCCURENCE DATA
  - DEGREE TO WHICH RISKS CAN BE CONTROLLED
  - ACTION TAKEN OR BEING TAKEN TO REDUCE RISKS

ISIS22

Contingency Planning

Elmer Bomlitz
Harris Devlin Associates

PRECEDING PAGE BLANK NOT FILMED

Notes

Notes

Information Security Program Development

James R. Wade

Battelle Memorial Institute

164-165-166

Notes

Notes

Investigating Computer-Based White-Collar Crime

Neal Findley
U. S. Secret Service

168 - 169 - 170

Notes

Notes

Trust: Formal Methods and Associated Techniques

Susan Gerhart

Microelectronics Computer Corporation (MCC)

172-173-174

Formal Methods for Trustworthy Systems

Susan Gerhart
MCC Software Technology Program
Austin Texas 78759
512-338-3492 gerhart@mcc.com

Topics:

- Characterize Formal Methods
- Survey applications
- Contrast international perspectives
- Assess research, applications progress
- Describe MCC Formal Methods projects

PRECEDING PAGE BLANK NOT FILMED

What are Formal Methods?

"Applied Mathematics of Software Engineering"

college sophomore through Ph.D. level

Use

logic, set and sequence notation,
finite state machines, other formalisms

In

- system models
- specifications
- designs and implementations

For

- highly reliable, secure, safe systems
- more effective production methods
- software engineering education

In various forms

guidance: structuring, partial specification

rigorous, formal
generated and worked proof obligations

mechanized: using proof assistants

What is a Formal Method?

Formal Method =

Specification Language

+

Methods

See: Jeannette Wing's "A Specifier's Approach to Formal Methods", in IEEE Computer, September 1990

Distinguishing feature – extensive mathematics

Specification Language

1. Constructs
   (a) Data
   (b) Control
   (c) Operations
2. Mathematical "semantics" (in principle)
3. Reasoning systems
   (a) Satisfies
   (b) Follows from
4. Communication Strategy

Methods

1. Modeling Principles
   (a) What goes in - events, objects, properties
   (b) Point of View
2. Specification Structuring
   (a) Composition – parts, properties
   (b) Generality – libraries, theories
3. Specification Analysis
   (a) Consistency and completeness checks
   (b) Validation
      i. On the right track? (review (client))
      ii. Omissions accounted for? (review)
      iii. Well expressed? (review)
      iv. Implementable? (review)
      v. Expected properties hold? (proving)
      vi. cases look right? (testing)
4. Design and Implementation Strategy
   - "data reification" mapping abstract to more concrete data
   - correctness-preserving transformations
   - Implement "traditionally"

All incur proof obligations for "satisfies" i.e. verification

Uses of Formal Methods

Requirements Analysis
Precision to reveal and remove vagueness, ambiguity, contradiction, incompleteness

Design
Expressing and checking module interfaces

Refinement
Expressing and checking satisfaction obligations

Documentation
Abstracting, organizing, and formalizing

Verification
Proving some property of some part of a system

Validation
Predicting and evaluating what the system will do, per the client wishes

Testing
Generating cases from specifications

Application Framework
Organizing pplication domain knowledge

Design Recovery
Extracting and evolving abstractions

Sample Applications in Progress

Project	Parties	Problem	Status
CICS	Oxford PRG IBM Hursley	Transaction Processing	Released, Measured (??)
Cleanroom	IBM FSD NASA SEL	Embedded, Restructurer	Released
ZEE	Tektronix	Oscilloscopes	Reports
Avalon/C++	C-MU	Atomicity	Preliminary
GKS, OA Doc.	British Standards Institute	Graphical, Documents	Published
SXL	GTE Labs	Protocols	In use
L.0	Bellcore	Protocols	In use
Anti-MacEnroe Device	Sydney Inst. Technology	Tennis Line Fault Detector	Report (Occam,CSP)
Security	Honeywell Ford Aero. Digital TIS ...	LOCK Multi-net Gateway Secure VMS Trusted Mach ...	In progress " " " "
VIPER	RSRE, Cambridge	Microprocessor Tools	Reports Newsletter
Verified Stack	CLinc	Microp, assembler, O.S.	Reports
Oncology	U. Wash.	Cyclotron	Starting
Reactor Control	Parnas, Ontario Hydro	Shutdown Certification	Reports, Certified
Murphy	U.C. Irvine	Safety	Reports
SACEM	French RR	Train Control	ICSE12

See FM89 proceedings, Springer-Verlag, to appear

International Contrast

Aspect	U.S.	Europe
technical	verification tools	precise methods
educational	theory, guilt	pragmatic skills
emphasized levels	code, models	specification
style	varied	model-based
environments	unintegrated	integrating
directions	security → breadth	breadth → safety
mandated start	1980	1990
funding drivers	NSA/DARPA	Alvey/Esprit
Bottom Line	Static	Progressing

Example applications discussed at VDM 90:

1. Hypertext reference model (Denmark/VDM)
2. Oscilloscope framework (Tektronix/Z)
3. Transaction system speedup (Norsk data/VDM)
4. Real-time system kernel (UK/Z)
5. Objective-C implemented CASE (UK/Z)
6. LaCOs starting (RAISE): Paris Metro, European Space, Bull OS, ...

MoD 0055/0056 - Regulatory Steps

MoD 0055 statement highlight:

17.1 Safety Critical Software shall be specified using formal mathematical techniques. A specification of the Safety Critical Software shall also be produced in clear English. Both specifications shall be included as part of the Procurement Specification. A list of mathematical techniques is given in Annex L.
Annex L: VDM, Z, OBJ, HOL, CCS, CSP, Temporal Logic, LOTOS

Possible Implications:

Context admission: safety-critical identification (0056 precedes 0055)

Commercial spill-over: Chemical, electrical industry regulations; EEC shift burden of proof to developers

U.K. competitiveness strategy: high-integrity systems

Alvey/Esprit commitment: industrialize formal methods

International climate: U.S. companies selling in Europe in 1992, Europeans could force upon international standards

Educational systems: Next generation well-trained in formal methods

Bottom Line - Research

Value established for

1. Identifying deficiencies, errors, discrepancies in new and old systems
2. Specifying medium-sized and non-trivial components, especially functional behavior
3. Gaining deeper understanding of large, complex systems.

Bounds are recognized (as in all engineering):

1. Informal to formal - requirements accuracy
2. Specifications as abstractions of real world
3. Many assumptions about the environment

Challenges

1. Non-functional behavior
2. Method combinations
3. More usable and robust tools
4. Specification libraries + domain theories
5. Scaling experiments
6. Integration in whole development process

Bottom Line - Applicability

Accepted into practice

Trusted Subsystems

Proofs, reviews, tests, analyses, qualified personnel,... for certification

Re-engineering

Gradual formalization of important modules and their documentation as part of evolution.

Standards

Formalization to achieve consensus and technical clarity

Potential

Exploration

Abstract experiments ala MCC's *SpecTra*

High Volume Subsystems

Verification plus test generation

Component Reuse

Generalization, high grade documentation, "recall" avoidance

New Products

Formal notation + informal methods

MCC Formal Methods Project

Goals:

1. Transfer to MCC participants
   (North American industry, government)
2. National testbed (tools, experiments)
3. Research on
   (a) New architectures for support technology
   (b) Integration of formal and informal material
   (c) High performance specification analysers
   (d) Novel validation methods

Specification prototyping + trustworthy systems

Work in progress:

1. Mechanized theory for temporal reasoning
2. Transition study (1 year from Sept. 1990, 10+ members)
3. Assembling testbed tools, doing experiments
4. CoDesign (hardware/software) exploration
   STP + CAD + outside
5. Proposals for government co-/funding

Some Key Research Questions

1. Define representations of issues (informal information) for:
   (a) System specifications: Methods, Languages, Environments
   (b) Client Communication
   (c) Validations : Proofs, Tests, Analyses, Reviews, Animations

   for various classes of applications and using MCC GERM technology
2. Identify basic technology to support a range of specifications:
   (a) State transitions
   (b) State and Object domains
   (c) Level and executability mappings

   emphasizing what is "executable" and the role of declarative techniques
3. Determine the usefulness of a forward-backward (proof-test) model of specification experimentation and validation
4. Evaluate animation in specification experimentation and validation
5. Evolve an architecture to accommodate extant tools

What is SpecTra?

1. Specification prototyping "shell"
   (a) Generic to languages (initially, ASLAN)
   (b) Generic to theorem provers (manage HOL, B-M initially)
   (c) Support for method, language, validation issue generation and analysis
2. Novel components
   (a) Issue/artifact nets to represent design records
   (b) Declarative language → Executable specifications
   (c) Parallel processing and abstract machines → High performance validation
3. Designed for exploration (a.k.a prototyping)
4. Support multiple modes of validation
5. Focus on system-environment boundary (versus code)

Testbed Description

1. Staffed primarily by academic visitors and industrial assignees
2. Source of target problems for the technology research
3. 1-2 year experiments on real but scaled down industrial problems involving 1-2 assigned personnel and 1-2 MCC researchers as advisors;...
4. Example Testbed projects:
   (a) defining and experimenting with an executable subset of international standard VDM using our flexible reasoning tools;
   (b) testing a formally stated standard for loopholes and inconsistencies and applying it to a real problem;
   (c) formally specifying and partially verifying a component library;
   (d) formally specifying some company's product using an issue base compiled during the product's (traditional) development;
   (e) jointly with another software engineering initiative, validating a communication protocol;
   (f) specifying and verifying a critical system component in parallel with a different organization using different technology;
   (g) investigating the reasoning requirements for a particular safety analysis technique.
5. Open to MCC's shareholders and associates and, possibly, European initiatives

Transition Study

Purposes:

1. Stand-alone assessment for decision-makers - FM or not?
2. Identify future directions for MCC FM project
3. Establish cooperative relationships with participants
4. Start-up funding for MCC project

Content:

- Fundamental Concepts
- Training and Instruction
- Modes of Use
- Major Applications
- Tools Survey
- Development Models
- Regulatory and Legal Trends
- Transitional Tips
- Research Needs and Strategy

Supported by experiments

Mechanics:

1. 1 Year, starting September 1, 1990
2. $60K for 10+ participants
   (a) Shareholders (STP/ACT and others)
   (b) Associates
   (c) New Associates ($7.5K program associate membership)
3. Delivery in videotapes, MCC meetings, final report

Technical

Secure Distributed Operating System and Verification

Doug Weber

Odyssey Research Associates

Theta — A Secure Distributed Operating System *

D. G. Weber, Joseph R. McEnerney, Rammohan Varadarajan
Odyssey Research Associates
301 A Dates Drive
Ithaca, NY 14850
(607) 277 2020

1 Introduction

The THETA project is an attempt to advance the state of the art in secure distributed operating systems. THETA (a Trusted HETerogenous Architecture) is an experimental secure distributed operating system being developed at Odyssey Research Associates, Inc. (ORA). (THETA was until recently called "SDOS".) The system is being designed and built to meet TCSEC B3 security and assurance requirements. An earlier phase of the project [SDOS 88a], [SDOS 88b] produced a design targeted towards a TCSEC A1 rating.

THETA borrows many of its concepts from Cronus, a distributed operating system developed at Bolt Beranek and Newman, Inc. [Cronus 86], [Cronus 88]. For example, the basic object-oriented client-server model of Cronus has been retained. The system architecture, however, has been redesigned to provide multi-level security, enhanced identification and discretionary access control, configuration security, audit, COMSEC protection and TCSEC assurance.

THETA is intended to support Command and Control applications potentially needed by the Air Force. This imposes several requirements on THETA. First, C^2 applications span many types of computer systems and require survivability, scalability and interoperability. Second, they involve diverse aspects of the use of secure information including collection, selection, aggregation, and analysis. Additionally,

*the work reported in this paper was supported by the Air Force Systems Command at Rome Air Development Center under contracts F30602-86-C-0146 and F30602-85-C-0056.

these applications involve monitoring and controlling physical devices that collect and use secure information.

The paper begins by discussing the goals of the project. The system design, architecture and security policy are discussed in sections 3, 4 and 5 respectively. We conclude by reviewing the current state of the project and the plans for future effort.

2 THETA Goals

An *operating system* provides abstractions by which users may use, share, and control the resources of an underlying machine. A *distributed operating system* (DOS) accomplishes this for resources at many locations that can be accessed by a network; it is therefore not a network, but rather built on top of one. A DOS presents the user with a uniform, location-independent interface even though the distributed resources may be heterogeneous. A *secure distributed operating system* is a DOS that provides access only if these are consistent with a security policy.

The goals of the THETA project are enumerated below:

- *Coherence and Uniformity*

  The THETA system should provide a coherent and uniform integration of the distributed processing resources. System services must be available to the user through a uniform set of abstractions. Objects such as files, directories, processes, services and I/O devices must be accessed using a global naming facility and a uniform set of communication primitives.

- *Heterogeneity and Evolution*

  Many distributed systems have evolved through the interconnection of existing stand-alone machines of possibly different hardware and software architectures. These machines may be connected by a local-area network (LAN) at a specific location or by a wide-area network connecting LANs at different locations. The THETA system should permit the interconnection of machines of differing architectures over different communication media in order to facilitate the sharing of information and computing resources between organizations, and to provide increased reliability and availability of services.

- *Reliability and Availability*

  The THETA system should be reliable in the sense that the integrity of its data should be maintained even across system failures. The THETA system should be available or be fault-tolerant so that services continue to be accessible even if parts of the system should fail.

- *Scalability*

  The THETA system may be configured with different processing elements to accommodate a range of users and specific applications. It should be possible to incrementally expand the system with additional resources over time.

- *Preservation of Existing Applications*

  The THETA system should permit the execution of existing applications such as compilers, editors, window systems, databases, etc. The design of the THETA system should not require the re-coding of these common applications. In addition, it should be possible to permit THETA users access to specialized computing resources that may be attached to the system such as high-speed parallel processors, special purpose symbolic processors, or high-speed graphics devices.

- *TCSEC Requirements*

  The THETA system is being designed to meet the TCSEC B3 functionality and assurance requirements. Therefore, the sharing of information and resources on THETA should be consistent with: the enforcement of a mandatory security policy; enforcement of a discretionary access control policy; reliable identification and authentication of users and their processes; and auditing of user and system activity. A trusted path will exist for security-critical operations.

- *Trusted Network Interpretation Requirements*

  The network interconnecting the components of the THETA system must provide message integrity, protection from compromise, and protection from denial of service, [TNI 87].

3 System Overview

THETA is an object-oriented [Jones 78] system. Objects are instances of abstract data types. The definition of a type includes the set of operations that are possible for objects of that type. There is a hierarchy of types. Each type with the exception of the root type, Object, has exactly one parent. A type may inherit operations from its parent. A type may also define new operations.

Objects can be accessed by invoking operations on them. Client programs act on behalf of the user to issue such invocations. THETA users interact with the system through the User Interface which permits execution of THETA system client or user-written application client programs. The invocation of an operation is the only way to access an object. Operations are implemented by object managers. A manager hides the internal representation of the object, and provides a precisely defined interface to the object. The kernel on the local host is responsible for locating a manager for the type, passing the invocation and returning the results. For this, the kernel may need to interact with its peers on remote hosts. All resources in the system are represented as objects, and all operations are carried out as described above.

Figure 1 illustrates the major system components and their relationships to each other.

The THETA Kernel is an multi-level process and therefore part of the MAC TCB. Several managers and clients may be multi-level too. The sections to follow will describe the system design and the major system components in detail.

3.1 Object Naming

THETA provides a global and location transparent naming facility to the user. A name is global if the name can be issued from any location and uniquely identifies an object. A name is location transparent if the location of the object is not directly encoded in the name itself. There are two levels of names for objects in THETA. The Unique Identifier (UID) is a machine-generated internal name, and the catalog name is an user-selected symbolic name.

THETA objects have a single UID which is stored with the object and is bound to the object at object creation time. The UID is not meant to be manipulated directly by users of the system. Its internal representation is optimized for handling by the machine. The UID includes the object's type, security label, and an unique number. Users typically want to reference objects using symbolic strings which are meaningful to them. The Catalog Manager provides a distributed and replicated service which maintains the mapping between user-defined symbolic names and system-maintained UIDs. The catalog is an hierarchical naming structure of the form ":a:b:c" where "a" and "b" are directories and "c" is a catalog entry in "b". Directories in this path name scheme are non-decreasing in security level. The catalog is distributed so that different hosts may manage different parts of the name space.

It is not required that every THETA object have a symbolic name. An object may have none, one or more symbolic names.

3.2 Object Replication

THETA provides reliability and availability by supporting replication of objects at multiple sites.

Objects which may be moved from host to host are called *migratory* objects. A

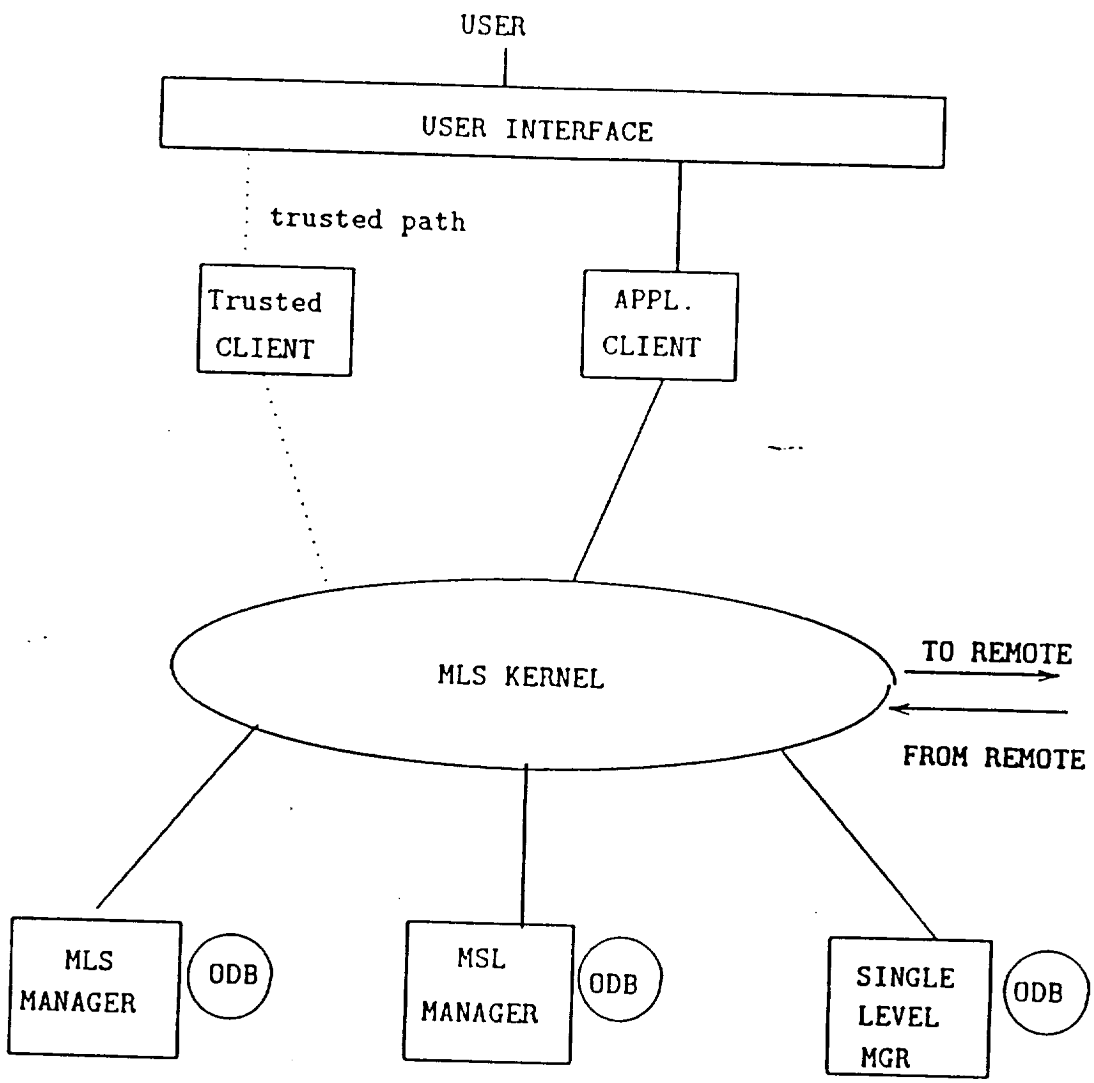


Figure 1: THETA System Components - Schematic

replicated object is one which has been duplicated and resides on more than one host. Each replica of the object has the same UID. The object may be accessed on any of the hosts where it resides.

Certain objects are *primal objects* which means that they cannot be replicated nor can they migrate.

THETA provides the necessary mechanisms to maintain consistency of replicated objects. The scheme used is a version vector scheme [Parker 83]. This classic problems of availability and consistency are resolved by allowing (at type definition time) read and write quorums to be set for each replicated type.

Replication of object managers is discussed in the next section.

3.3 Object Managers

An object manager is a process that maintains the representation for all objects of a given type (in an object database — ODB) and implements the operations that are supported on objects of that type.

In THETA, managers only exist for the types corresponding to the leaves of the type hierarchy. A manager may manage objects for one or more of these types.

Managers, like the objects that they manage, can be replicated. However, since each manager is a process, and processes are primal objects, the replicated managers have unique UIDs. Also, there cannot be more than one manager for a given type, at a given security level on a single host. The managers use the version vector mechanisms to maintain consistency of the objects that they manage.

THETA managers also provide for asynchronous processing of several invocations concurrently. For this, a task abstraction is supported and library support for a multi-tasking environment is made available.

3.3.1 Single-Level and Multi-Level Managers

A manager may be a single-level object manager or an MLS object manager. A single-level object manager only manages objects at its security level and is implemented as a single-level process.

A multi-level object manager is one which can handle operations on types that it manages at a range of security levels. A multi-level manager may be designed as a single multi-level secure (MLS) manager process or multiple single-level (MSL) manager processes. If it is implemented as a MLS process, then the manager is part of the mandatory TCB and is trusted to perform mandatory access checks.

Should a multi-level service be MLS or MSL? There is a fundamental difference between the two approaches. The MLS manager is trusted, it can, therefore, enforce a security policy different from that of the constituent operating system (COS) (see section 4 for details of THETA architecture). The MSL managers are bound by the COS's security policy. Numerous other considerations like system resource and performance constraints, TCSEC assurance requirements, etc. will affect the approach. The decision is best made on a manager-by-manager basis.

3.3.2 System Managers and Application Managers

There is another metric for classifying managers in THETA — the type of object managed. As explained in section 3, THETA types can be classified into system types and user types. Correspondingly, there are system managers and user-written (application) managers.

An user-written application manager manages only user-defined types and is not permitted to manage THETA system types. These manager may be single-level managers or multi-level managers. If an user-written manager is to be multi-level, it can be constructed using the MSL scheme without any special procedures, because this construction does not extend the THETA mandatory TCB. On the other hand, if the manager were to be MLS, it must go through a certification processes before it can be admitted into the system.

System managers are part of the basic THETA system. Since these types are fundamental and will be used by clients at all levels, THETA system managers will be a multi-level service. Like the user-written managers, multi-level service offered under the MLS scheme must have undergone the necessary certification procedures.

3.3.3 Tools for Manager Generation

THETA provides the user with a set of tools for manager generation. The user defines a type and a manager using the type-definition and manager definition languages. He then uses the manager generation tools to build a skeleton of an object manager. The skeleton takes care of message packing and unpacking, conversion from canonical to internal representations and vice-versa, mandatory and discretionary access checks that may be necessary for an operation, etc. The user has only to fill in code for the type specific operations that the manager has to support.

3.4 Clients

A client process is one that interacts with THETA on behalf of an user. (The interaction is achieved by invoking operations on objects.) More importantly, unlike managers, clients do not support operations on abstract data types. Most clients are untrusted user written application programs. Some clients, however, may include trusted software which has been demonstrated to be free from Trojan Horses and can truly reflect the user's intentions.

3.5 Principals and Groups

Every THETA user or manager has a principal name which is stored in a corresponding principal object. Managers have principals names which correspond to the name of the manager. The principal object is managed by the Authentication Manager which may be replicated. Every principal object contains a list of groups to which the user belongs. When the user logs in, a default group is enabled and becomes active. There may be groups to which the user belongs that are not enabled automatically. Every group object contains a list of the principals that belong to that particular group.

THETA principal and group names are global and are used by managers to perform DAC checks.

3.6 The THETA Kernel

The THETA Kernel is a multi-level entity and is part of the mandatory TCB. The major Kernel components are the Host Manager (that manages operations like startup, shutdown, etc.), the Process Manager (that manages the process table) and the Switch (which routes messages). The Kernel intercepts all communication between THETA processes. The protocol used for local communication is the Operation Protocol. In the following sections we will elaborate on the Switch and the Operation Protocol.

3.6.1 Operation Switch and Locator

The THETA Switch is responsible for routing operations from clients to the correct object manager. This routing is based on the object's UID. The Switch is composed of a *Locator* and an *Operation Switch*. The Locator determines the host location of the object. If the object is of primal type, then the invocation must be routed to a manager on the local host. If the object is not primal and if the object is not present locally, then the Locator must determine the host location of the object. The object's location may be present in a local cache. If there is a miss on the cache, the Locator performs a Locate operation on the generic object of the type using the network's broadcast mechanism. All managers which have a copy of the object will respond positively to the Locate.

Once the object is located, the Operation Switch routes the operation to the Switch on the appropriate host. The Operation Switch maintains IPC connections with all local clients and managers and network connections with remote Switches. It also listens for request for new connections either locally or from remote hosts.

3.6.2 Operation Protocol

The Operation Protocol is used by clients and managers for communications with the Kernel. The basic inter-process communications (IPC) primitives are:

- Invoke — Invoke is used to invoke an operation on an object. A manager handling an invocation may need to perform secondary invocations on other objects (possibly of different type) to complete the primary invocation.
- Send — Send is used by managers to send a message (response) directly to the invoker.
- Receive — Receive is used by both managers (to get the next invocation or response from a secondary invocation) and clients (to receive the reply to a primary invocation).

4 THETA Architecture

THETA is implemented using a layered architecture. The THETA clients, managers, and Switch are implemented on top of an existing secure Constituent Operating System (COS). A process becomes an THETA process by interacting with the Kernel via the Register Process protocol. An important goal of the design is that THETA be implementable without modifications to the COS and implementable on a system of heterogeneous COSs. A COS must meet TCSEC B3 security and assurance requirements for the THETA system to be B3. The following features of the COS are used:

- assured process separation — direct interprocess communication that is not controlled by the system must be disallowed. To achieve this the MAC, DAC and user and process identification mechanisms of the COS will be used.
- non-interference with process operation — THETA processes responsible for security must not be tampered with. The same COS mechanisms as in the previous bullet are used.

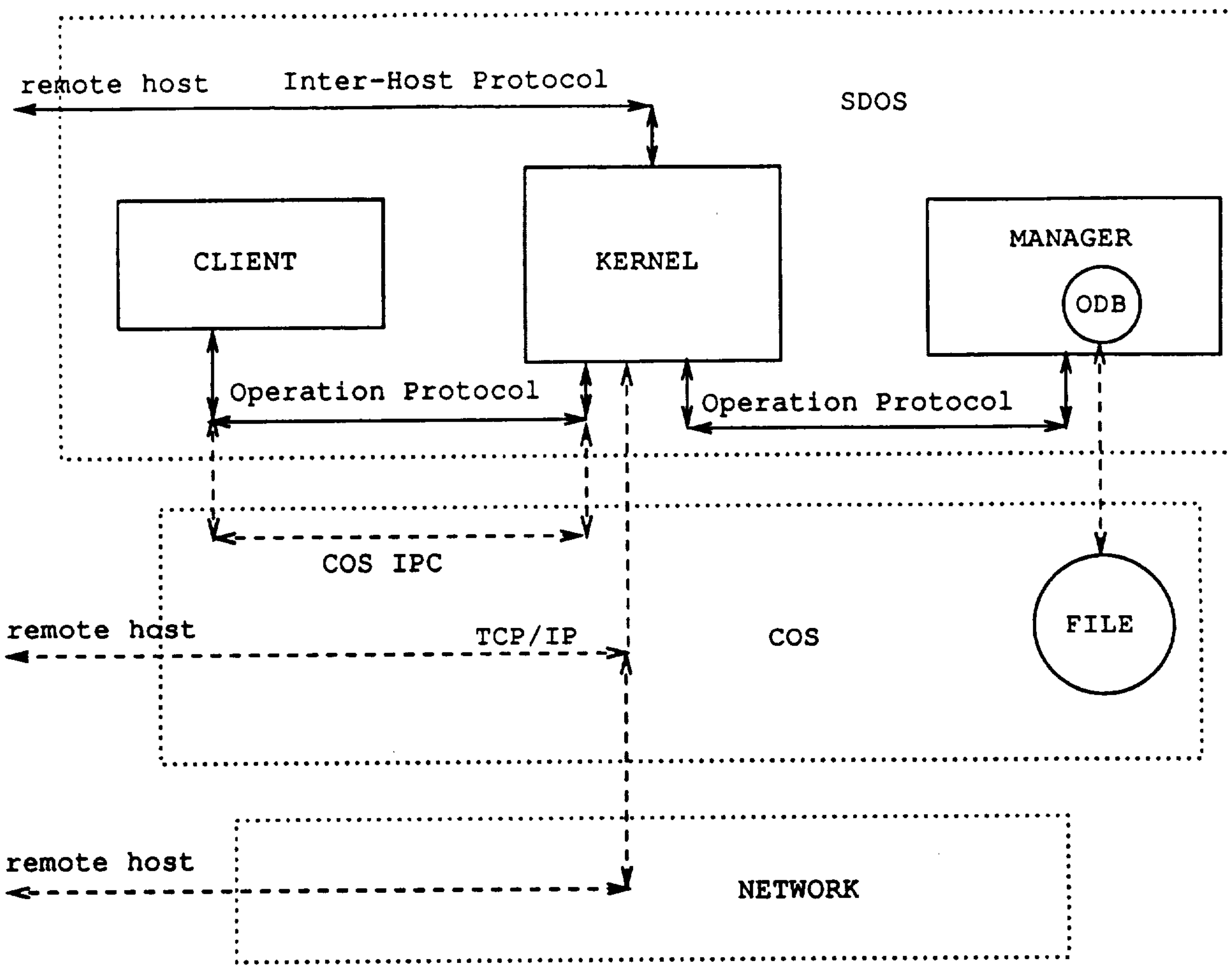


Figure 2: THETA Layering

- stable storage: data needed for enforcing security and for maintaining object representations must be protected. The COS file system will be used to achieve this.
- IPC support — trusted path, local IPC and TCP/IP facilities of the COS are used to support THETA IPC primitives and protocols.
- device support — COS device drivers are used for device support.

Figure 2 illustrates the layering architecture for the THETA implementation.

5 Security Policy

The security policy for THETA can be grouped into the following: (for details see [Proctor 89] and [SDOS 88a])

- A discretionary access control (DAC) policy, designed to restrict the use of abstract operations based on client identities.
- A mandatory access control (MAC) policy controlling the flow of information based on security levels.
- A configuration policy to define the security configuration.

The MAC and DAC policies are clearly separated. In fact, they operate at different granularities in the object model. The mandatory policy refers to the message passing structure which is used to implement the object model. The discretionary policy is stated in terms of abstract operations of the model. The policy is a global one, stating constraints for the entire system rather than for individual hosts.

The MAC and DAC policies are discussed in further detail in the following sections.

5.1 MAC Policy — "Restriction"

A distributed operating system mandatory policy must be defined in terms of message passing between active entities, rather than the traditional Bell and LaPadula read and write operations of an active entity on a passive entity.

The THETA MAC policy has two components:

- rules for message passing — these prevent direct downgrade of information.
- a policy for each multi-level entity — this prevents compromise of information via covert channels.

The multi-level security policy was based on an emerging theory of information flow security being developed at ORA. This theory defines information flow in terms of deductions that can be made about unseen (higher security level) events in a system's history. The policy due to McCullough, called "restriction", [ORA TR88], [McCull 87] was chosen. Under a restriction policy, a system component is secure provided it does not allow information to flow from high security levels to lower ones. Restriction also has the additional property of *composability*: two subsystems having that property can be hooked together to form a larger system also having the property. The THETA policy requires that the THETA TCB be restrictive. Since restriction is a composable property, it is sufficient to demonstrate that the components of the TCB are restrictive. The fact that security verification can be decomposed in this fashion is a tremendous advantage when trying to build a distributed secure system such as THETA. Composability can also be exploited to add multi-level services and hosts to a distributed system in a secure manner without the need for re-verification of the entire system.

Our work on the Phase I effort developed techniques for demonstrating compliance with restriction using the Gypsy Verification Environment [Weber 87].

5.2 DAC Policy

Since object managers are the entities that support operations on objects and DAC restricts operation executions, all THETA system managers enforce discretionary access control on their objects. An Access Control List (ACL) is maintained for each object which indicates which users may perform which operations on that object. The DAC policy is necessarily object-dependent since operations and their semantics vary with the type.

6 Current Status

The present THETA project is a thirty month effort ending in early 1991 with a demonstration of the prototype system.

The current system is aimed at providing support for the connection of multiple THETA hosts on a single local area network. In addition, single level untrusted hosts may be attached to the network using an MLS THETA acting as a front-end access machine.

ATT Sys V MLS UNIX has been selected as a representative COS. Two hardware architectures — the 6386 PCs and 3B2 will be supported.

The system requirements [SDOS 89a], architecture and security policy [SDOS 89b] documents have been completed. The formal security model and the detailed design document are in preparation. Trial implementations of key design features are also underway.

7 Plans for future effort

The system will evolve to permit the connection of multiple THETA machines over an open Internet. THETA will provide the necessary network protection required for the transmission of multilevel data. (This must include encryption.) Untrusted single-level Cronus hosts may reside on the same network. Communications between untrusted Cronus hosts and THETA hosts will be accomplished by using an THETA host as a gateway.

There is considerable activity in the commercial arena on developing secure operating systems. We hope that THETA will encompass these secure platforms as they become available.

The security policy that is afforded by THETA will be relaxed to accommodate violations that are permissible in normal C^2 systems. Current efforts at ORA hold promise [Sutherland 89].

Support for transaction oriented processing to help host DBMS must be made available in THETA.

References

[Cronus 86] Schantz, R., Thomas, R., and Bono, G. *The Architecture of the Cronus Distributed Operating System.* Proceedings of the IEEE 6th International Conference on Distributed Computing Systems, May 1986.

[Cronus 88] *Cronus: Revised System/Subsystem Specification.* BBN Report No. 5884, Revision 1.6, August 1988.

[Jones 78] Jones, A. *The Object Model: A Conceptual Tool for Structuring Software.* Operating Systems, An Advanced Course; Lecture Notes in Computer Science, Springer-Verlag, Editors: Bayers, Graham, and Seegmuller, 1978, pp. 8-16.

[ORA TR88] McCullough, D. *Ulysses Security Properties Modeling Environment: The Theory of Security.* Odyssey Research Associates, July, 1988.

[McCull 87] McCullough, D. *Specifications for Multi-Level Security and Hook-Up Property.* Proceedings of the 1987 IEEE Symposium on Security and Privacy, April 1987, pp. 161-166.

[Parker 83] Parker, G.S., et al., *Detection of Mutual Inconsistency in Distributed Systems.* In IEEE Transactions on Software Engineering, Volume SE-9 (3) 240, May 1983.

[Proctor 89] Proctor, N., and Wong, R. *The Security Policy of the SDOS Prototype*. To appear in Proc. 5th Aerospace Computer Security Conference, December 1989.

[SDOS 88a] Vinter, S.T., et al., *The Secure Distributed Operating System Design Project*. RADC-TR-88-127, Jan. 1988.

[SDOS 88b] Casey, T., et al., *A Secure Distributed Operating System*. Proceeding of the 1988 IEEE Symposium on Security and Privacy, April 1988, pp. 27-38.

[SDOS 89a] ORA Staff, *System Segment Specification for the SDOS*. ORA Technical Report, TR 25-1. Feb. 1989.

[SDOS 89b] ORA Staff, *System Segment Design Document for the SDOS*. ORA Technical Report, TR 25-2. June 1989.

[Sutherland 89] Sutherland, I., Perlo, S., and Varadarajan, R. *Deducibility Security with Dynamic Level Assignments*. In Proceedings of the Computer Security Foundations Workshop II, June 1989.

[TCSEC 85] *DoD-5200.28-STD, DoD Trusted Computer System Evaluation Criteria*. December 1985.

[TNI 87] *NCSC-TG-005, DoD Trusted Network Interpretation*. July 1987.

[Weber 87] Weber, D.G., and Lubarsky, R., *The SDOS Project — Verifying Hook-up Security*. In Proc. 3rd Aerospace Computer Security Conference, December 1987.

[Wong 89] Wong Ray, et al., *The SDOS System*, To appear in the 12th National Computer Security Conference, October 1989.

Trusted Ada

John McHugh
Computational Logic Inc.

TRUSTING ADA

John McHugh
Computational Logic, Inc
Durham, North Carolina
15 MAY 1990

OVERVIEW

- DoD 1 to Ada-83
- The Perils of Complexity
- Interim Solutions
- Ada 9X
- Research Directions

Early '70s at DoD

- A 1973-74 study showed DoD spent $3 billion/year on software, 50% for embedded systems.
- Typical program is 10^5 to 10^6 lines, has a staff of 50+ programmers, and a 10-15 year lifetime, longer than its hardware base.
- Programs are life- and safety-critical.

CLI

Trusting Ada

Early '70s at DoD, continued

- Software costs were absorbing an increasing portion of system costs, mostly in maintenance.
- Much of the work was so specialized that only the original developer could support it.
- Much of the cost and specialization was attributed to use of obsolete programming languages lacking support for "modern" software engineering methods
- Many projects used their own languages or dialects.

CLI

Trusting Ada

The HOLWG

- 1975 Higher Order Language Working Group: DoD, Services, DCA, NSA, DARPA
- Choose a standard military language (DoD-1 working name) for all new software
- Language could be an existing or new one, but it would have to be worthy of recognition as *the* standard real-time programming language.
- Research funding for unrelated efforts was halted.
- Five existing languages approved for interim use.

Trusting Ada

Strawman to Tinman

- April 1975–Sample requirements, more for style than content, issued by HOLWG. These were termed "Strawman".
- August 1975–"Woodenman" requirements issued as tentative language requirements.
- January 1976–Official requirements, "Tinman" issued.
- Progress of requirements development involved broad participation from avionics, guidance, command and control, and simulation communities.
- To everyone's surprise, all had essentially the same requirements, which also covered scientific and commercial applications as well.

Trusting Ada

Toward a New Language

- Summer 1976–Twenty-three existing languages evaluated against Tinman.
  - FORTRAN, COBOL, PL/I, Algol 60
  - HAL/S, TACPOL, CMS-2, CS-4, SPL/I, JOVIAL J3B, JOVIAL 73, CORAL 66
  - Algol 68, Pascal, Simula 67, LISP, Euclid, EL1
  - LTR, RTL/2, PDL/2, PEARL, MORAL
- September 1976–Workshop on the feasibility of a language to satisfy Tinman.
- January 1977–All candidates found lacking. New language feasible and desirable. Pascal, PL/I or Algol 68 could be used as basis.

CLI

Trusting Ada

The Design of a New Language

- April 1977–Tinman reorganized as "Ironman" language specification and proposals sought for languages based on Pascal, PL/I, or Algol 68.
- Evaluation criteria: reliability, maintainability, efficiency.
  - Reliability–Language rules to catch errors before run time, constructs for good programming practice, constructs for reuse.
  - Maintainability–Ease of reading even at the expense of writing.
  - Efficiency–Fast compile, compact object code, fast run time.

CLI

Trusting Ada

Three Phases

- Phase 1: July '77-January '78

  4 Prototype Languages

- Phase 2: April '78-April '79

  Requirements modified to become "Steelman". 2 languages remain.

- Phase 3: June '79-June '80

  Green language chosen, named Ada

9 Trusting Ada

CLI

Phase I

July '77-January '78

- 4 prototype languages from 17 proposals (all but 1 based on Pascal)
  - Green-Ichbiah, CLI Honeywell Bull
  - Red-Ben Brosgol , Intermetrics
  - Blue-John Goodenough, SofTech
  - Yellow-Jay Spitzen, SRI

10 Trusting Ada

CLI

Phase II

April '78-April '79

- Requirements modified to become "Steelman". Reference manual, rationale, test translator, and formal semantic specification for 2 languages
  - Green and Red languages evaluated

Phase III

June '79-June '80

- Green language chosen, named Ada
  - Preliminary Ada manual, June '79
  - Revision under HOLWG and distinguished reviewers
  - Proposed standard, July '80

Toward Ada 83

- Language accepted by HOLWG in August 1980
- Implementers urged to start work
- HOLWG becomes AJPO (Ada Joint Project Office) in December '80. MIL-STD 11818-A adopted. Ada registered as a trademark
- Language proposed as ANSI standard. Expectation of minor suggestions for manual changes wrong.

Hints of Trouble

- Proposed standard received 400 pages of comments, 750 questions.
- Both editorial and language-design issues raised.
- Many objectors thought that the language was too complex
- Manual overhauled but definition of a "Legal Ada Program" and its meaning changed little

MIL-STD 1815-A and Ada-83

- Second review July-October 1982
- Additional minor changes
- January 22 1983: MIL-STD 1815-A adopted
- ANSI standard, February 17, 1983

15 Trusting Ada

Barriers to Trust

Despite the initial requirements and the development process, there are a number of barriers to the development of Trusted Ada.

- The size and complexity of Ada
- The size and complexity of Ada runtime systems
- The lack of support for reasoning about Ada Programs

16 Trusting Ada

Size and Complexity

Ada is a large language with a complex and imprecise reference manual.

Since 1983, over 1200 separate requests for clarification or interpretation have been filed with the language maintainers.

- •800 commentaries (AIs)
  - 300 presentation (typos, format)
  - 500 substantive
    - 50% static semantics (compile time)
    - 50% dynamic semantics (run time)
- •200 approved by AJPO

17 Trusting Ada

CLI

Features Interact

- Overloading
- Separate Compilation
- Private types
- Signals and handlers
- Tasking
- Optimization and code generation

18 Trusting Ada

CLI

Incomplete Definitions

- Ada leaves a number of freedoms to the implementor
  - Orders of evaluation
  - Parameter-passing mechanisms
- Ada defines forms of erroneous execution
  - Neither compiler not runtime system is responsible for detecting these
  - Effect of the program is undefined

CLI

Trust and Predictability

- Any trusted system, whether security- or safety-critical, must have predictable behavior.
- The combination of feature interaction and erroneousness makes this very difficult

CLI

Verification Support

- Proofs of program properties are one way to obtain high levels of assurance
- Support for verification in terms of both a formal language definition and an assertion mechanism were Steelman requirements
- The requirements were not satisfied by Ada-83, largely due to personalities and the complexity of the language.

Run Time Issues

- Most Ada implementations rely on a large, complex, run time support system.
  - Tasking implementation
  - Storage management
  - I/O
  - Host system interface
  - Operations on complex types
- The trustworthiness of this code is difficult to establish

Trusted Ada Applications

- The previous discussion is cautionary
- Trusted applications have been built in Ada
- The trick seems to be use of a restricted subset of Ada

ASOS: An Example

- ASOS-"Army Secure Operating System"
- Multi-Level Secure-A1
- The ASOS operating system was developed using a subset of Ada dictated by a variety of security-related factors. A significant factor is the ASOS use of Gypsy specifications for its FTLS and the need for showing correspondence between the Gypsy FTLS and the Ada code.

The ASOS Subset

- Ch 2 - Minimize usage of floating point
- Ch 3 - Initialization of all variables except access types
  - NEW operation not used (no heap in ASOS kernel)
- Ch 5 - GOTO not used
- Ch 6 - No aliasing, no functions with side effects or references to globals
- Ch 8 - No renaming declarations
- Ch 9 - No tasking in the kernel. ASOS provides tasking support for applications

Trusting Ada

The ASOS Subset, continued

- Ch 11 - Exceptions are not propagated across the TCB boundary. They must be locally handled and explicitly propagated at routine boundaries elsewhere.
- Ch 12 - Use of generics is limited in ASOS
- Ch 13 - System-dependent features restricted. ASOS runs on bare hardware
- Ch 14 - ASOS minimizes use of Ada I/O

Trusting Ada

Subset Discussions

- By taking a conservative approach, trusted systems can be built in Ada. ASOS uses almost none of the standard run time support.
- It may be necessary to look closely at generated code and run time usage to avoid, for example, heap allocation in a long-lived system
- The current state of compiler practice is improving, but compiler and run time issues must still be considered on a case-by-case basis

27 Trusting Ada

CLI

Ada 9 X

Ada is undergoing revision. The trusted systems community has raised a number of issues in connection with this revision. They are summarized in the following slides.

28 Trusting Ada

CLI

Requirement A

IDENTIFY AND *JUSTIFY ALL* ELEMENTS OF THE STANDARD THAT PERMIT UNPREDICTABLE PROGRAM BEHAVIOR.

e.g., Program blockage
Integer (1.5) $\stackrel{?}{=}$ Integer(1.5)

INTENT IS TO ELIMINATE WHERE POSSIBLE AND FORCE ANALYSIS AND COST BENEFIT DECISION ELSEWHERE.

REQUIREMENT A -continued

1) Eliminate most erroneous cases
2) Eliminate "incorrect order dependency"--define order-dependent semantics
3) Define undesirable implementation dependency (UID)
4) UID has defined effect, not cause for "program error"
5) Implementations shall attempt to detect remaining erroneous and UID cases
6) Specific cases of undefined variables:
   a. Majority - URG position on LHS usage
   b. Minority - catch <u>all</u> usage

REQUIREMENT B

EXPOSE IMPLEMENTATION CHOICES

1) Language choices (LRM alternatives)
2) Implementation strategy (storage management, scheduling, etc.)
   - Static choices
   - Dynamic choices
   - What can user control?
   - How can information be shared with others? With tools?

   Choices include:

   a) Parameter passage

   b) Optimization

   c) Heap vs stack vs ...storage management

CLI

REQUIREMENT C

ALLOW USERS TO CONTROL

IMPLEMENTATION TECHNIQUES

Certain implementation choices lead to explosive growth in possible execution behaviors.

Implementations must honor–or reject with warnings–user directives for items such as parameter passing mechanisms, orders of evaluations, etc.

This is analogous to the representation specification for data.

CLI

REQUIREMENT D

IMPLEMENTATIONS SHALL ATTEMPT COMPILE OR RUNTIME ANALYSIS FOR KNOWABLE INSTANCES OF UNSOUND PROGRAMMING AND ISSUE WARNINGS/EXCEPTIONS AS APPROPRIATE.

- Aliasing
- Unsynchronized sharing
- Uninitialized variables
- Etc.

REQUIREMENT E

PROGRAM BEHAVIOR TO BE DEFINED OR PREDICTABLE IN THE FACE OF OPTIMIZATION

We call for further study on the following

- Canonical order of evaluation vs radical optimizations
- Exceptions
- Side effects
- Possibility of pragma control

REQUIREMENT F

FORMAL STATIC SEMANTICS AS PART OF
ADA 9X STANDARD

The formal definition to be accompanied by tools that facilitate use for answering questions about the legality and meaning of programs.

While this does not necessarily change the language, development of the definition and tools may contribute to language changes.

N.B. Parameterize formal definition for implementation decisions and architecture/environment.

CLI

Trusting Ada

REQUIREMENT G

DYNAMIC SEMANTICS AS ONGOING EFFORT WITH
AIM OF INCORPORATIONS IN NEXT STANDARD.

This area has enough uncertainty to keep it off the Ada 9X critical path. On the other hand, development of portions of the dynamic semantics as part of the Ada 9X effort should aid in evaluating and understanding proposed language changes.

N.B. Parameterize formal definition for implementation decisions and architecture/environment.

CLI

Trusting Ada

REQUIREMENT H

ASSERTIONS

MAJORITY

1) Need dynamic semantics for assertions to be useful for proof
2) Suitable form not known
   - Extend Ada expressions
   - Ada vs spec functions
   - Etc.

   ∴ Wait, but work on issue

MINORITY

1) Anna exists
2) Anna is better than nothing

   ∴ Use Anna for now

DON'T PRECLUDE LATER CHOICE/DECISION

Research Issues and Efforts

- Ada 9X will not have a full formal definition.
- Research is underway that addresses some of the problems affecting the use of Ada in trusted systems
  - AVA at Computational Logic, Inc.
  - Penelope at Odyssey Research Associates
  - ANNA at Sanford
  - Low Ada at NPL
  - The Ada 9X Language Precision Team

AVA

AVA (A Verifiable Ada) is a DARPA-funded effort to formally define a subset of Ada

- Subset is not unlike the ASOS subset
- Manual (derived from Ada manual) exists
- Formal definition based on IRIS abstract syntax for AVA is written in Boyer-Moore logic

Penelope

Verification system for Ada subset developed by Odyssey Research Associates.

- Subset is somewhat more restrictive than ASOS or AVA but expanding
- System in limited experimental use

Trusting Ada

Penelope

Verification system for Ada subset developed by Odyssey Research Associates.

- Subset is somewhat more restrictive than ASOS or AVA but expanding
- System in limited experimental use

CLI

ANNA

Specification language for Ada in the form of stylized comments.

- Attempts to cover full Ada language
- Tools produce executable, run time tests
- Integrated with Verdix compiler
- Available for the asking
- Handbook in preparation

CLI

Low Ada

A proposed low-level intermediate language for Ada compilation.

- Simple static semantics
- Should be easy to provide dynamic semantics and proof rules
- Not yet implemented

Language Precision Team

PRDA issued by Ada 9X project.

- Supports Ada 9X mapping team by providing formal analysis of selected language topics
- "Creeping formalism" approach to demonstrating utility of formal methodology
- May have some influence on Ada 9X language

A Conceptual Model for Supporting B3+ Dynamic Multilevel Security and Integrity in the Ada Runtime Environment

Charles W. McKay
University of Houston-Clear Lake

A Conceptual Model for Access Controls to Support Dynamic, Multilevel Security & Integrity (DMLSI) in the Run Time Environments (RTE) of Large, Complex, Nonstop, Distributed Systems

Charles W. McKay, Director
Software Engineering Research Center
High Technologies Laboratory
University of Houston-Clear Lake
(SERC/HTL@UHCL)

Purpose of these notes :

To facilitate an understanding of :

- the major issues &
- the major segments & relationships of a proposed solution architecture such that this understanding :
  - `scales down' from `B3+' issues & solutions at a conceptual level appropriate for the above types of systems to smaller, simpler, & less demanding/critical systems &
  - facilitates mappings from the conceptual to implementation models.

Security. "The protection of computer hardware and software from accidental or malicious access, use, modification, destruction, or disclosure. Security also pertains to personnel, data, communications, and the physical protection of computer installations." (IEEE, 1983) Note that in the DOD Trusted Computer System Evaluation Criteria, (1983) (also known as the 'Orange Book'), security focuses primarily upon protection of classified data. In CLAR and CLAD, security is a complement to, but not a substitute for, integrity (q.v.).

Integrity. The correctness of an aspect of the system. "Resistance to alteration by system errors." (Oxford, 1983)

Security Kernel/Integrity Kernel. Mechanisms in the lowest layer of a virtual machine (q.v.) that supports policies for security/integrity. Note that a mission and safety critical (MASC) kernel (q.v.) must contain mechanisms to support security and integrity simultaneously.

Quality. "The totality of features and characteristics of a product or service that bears on its ability to satisfy given needs. (ANSI/ASQC A3-1978)" (IEEE, 1983)

Reliability. "The ability of an item to perform a required function under stated conditions for a stated period of time. (ANSI/ASQC A3-1978)" (IEEE, 1983) See also the definitions of 'software reliability' and 'system reliability'.

Safety. "The probability that a system, including all hardware and software and human-machine subsystems, will provide appropriate protection against the effects of faults, which, if not prevented or handled properly, could result in endangering lives, health, property and environment." (McKay, 1987)

A View of the Problem Space:

Three Distinct & Dynamic Management Domains

Mgt.* of :
Subject Objects & their :
- Views (sets of capabilities)
- Roles (view augmentations)
- Changes in Views & Roles

Mgt.* of :
Target Objects & their :
- access rights for
  - Views
  - Roles
- Changes in these Acc Rts.

System Mgt.* of Shareable Services & Resources

Some Relevant Definitions

- Objects
  - instances of abstract types
  - communicate by messages only
  - have an Abstract I/F Spec (AIS) part
    (=> contexts of operation &, for each context
    - services & resources, to be provided, affected, or used
    - how well these are supported
    - under what circumstances)
- And an encapsulated implementation part
  - May be
    - active (own thread of control)
    - passive (borrows threads of control & has none of its own)
    - neutral (neither owns nor borrows a thread of control)

- RTE Enforcement of Access Controls for DMLSI :
  - run time constraints on the interactions of subject objects, target objects, & system services & resources. The constraints are enforced by RTE mechanisms that support the DMLSI policies for the system & its applications.
- Other Relevant terms & Concepts
  - Conceptual Model of a Solution Architecture : An Approach to Mapping to Implementation Models
  - Mandatory Access Controls
  - Discretionary Access Controls
  - Basic User Capabilities, Roles, & Adoptions
  - Addressing with capabilities & intents; Access checking against access control lists & denials

	R's	A's of R's	Tgt Objs	A's of Tgt Objs
Existence Read Write Append Execute Cntrl Access Rts All				

SERC/HTL@UHCL

- **Solution Architecture Satisfies Specs For :**
  - **logical properties of Conceptual Model**
  - **physical properties of Implementation Model**
  - **Mapping of C.Model to I.Model**
    - Stable I/F Sets (Integrated Cfg. Items in Deployed & Operating Sys that satisfy S I/F Set criteria)
    - Stable Frameworks (Cfg Items that satisfy SF criteria)
    - Virtual I/F Sets (composed of AIS's that satisfy criteria for abstract types, objects, & messages)
    - Precise Modeling Support* in EA/RA Form
    - Entities

Model. "(1) A form in miniature; ...
(2) A generalized, hypothetical description, often based on analogy, used in analyzing or explaining something;" (Webster, 1972)

A *representation at one or more levels of *abstraction of a set of *concepts for a set of real world processes, products and/or *interfaces.

Entity-Attribute/Relationship-Attribute (EA/RA) Models. Instances of *models of some portion of a problem space or a solution space expressed in EA/RA form. These may also include instances of models which depict the mapping of:

- some portion of a problem space to a corresponding portion of a solution space
- some portion of an *abstraction at one level to a corresponding portion of abstraction at another level.

Formal Method. Consists of, or incorporates, a formal description technique (q.v.) (BCSWG, 1990) "Mathematically based *method for the *specification, *design and production of software. Also includes a logical inference system for *formal proofs of correctness, and a *methodological framework for software development in a *formally verifiable way." (MOD 0055, 1989)

Formal Model. A *model having a sound mathematical basis which is used to meet and exceed the C3FTC criteria for the *semantics of *precise models by allowing *formal verification via *proofs of correctness. Formal models contain no ambiguity for all legal sets of *states, stimuli, and the effects of *state transformations. Formal models facilitate the use of *formal methods. (The C3FTC criteria for the semantics of precise models are measures for: *consistency, *completeness, clarity, feasibility, testability, and correct operation while deployed in its *target environment.)

Precise Models. "Precise models have defined *semantics for all *entities of the *model and their *attributes, and for all *relationships among the entities and their attributes such that the C3FTC criteria are satisfied for all defined sets of legal operations upon legal values within legal *contexts." (McKay, 1988) See C3FTC criteria described in 'formal models'.

Conceptual Model. Describes the *architecture of a *design solution to a complex problem or *class of problems. It presents the major *segments, *attributes for those segments, major *relationships, and attributes for those relationships at a sufficient level of *abstraction necessary to understand and control complexity in evolving mappings to a working implementation *model. In *CLAR and *CLAD, these segments and relationships are abstractions of explicit design decisions that have been evaluated and selected because of a balance of considerations of *risk management, sequencing of development dependencies, *modularity, and opportunities for parallel development activities.

Meta Info for Subject Objects

- **Context Info**
  - Model Reference (eg. Schema, Dictionary)
  - User/Group/Project Id
  - Clearance Level (eg. secret, top ...)
  - Location & Device Id (eg. secure terminal ...)
  - Password (Optional)
  - Unique ID of Thread / Transaction / Subtransaction
  - List of currently adopted roles
  - Priority/Time/Other Constraints
  - Capabilities
- **Info on Services & Resources for Tgt Obj**
  - Logical Reference (may be alias)
  - Type (May be Model Ref.)
  - Unique ID (i.e. system)
  - Intent(s)

Meta Info for Target Objects

- Once for each object
  - Context Info
    - Model Ref. (eg . Schema, Dictionary)
    - Unique Id
    - Ownership
    - Classification Level (eg. secret, top ...)
    - Multiple copy References (Number, where, ...)
    - Access History
    - Encryption Info (Optional)
    - Mgt.Info for Priority / Time / Other Constr.
    - Mgt.Info for Access Control (eg. locks, multiple copies, ...)
  - Multiple Instance Info (As needed)
    - Context Info
      - Logical reference
      - Access Id for Authorized User(s) / Group(s) / Project(s)
      - Denials List
      - Location & Device Restrictions (Optional)
      - Password (Optional)
      - Template Restrictions (if any)
    - Services & Resources
      - Access Rights List

Technical

Complexity Issues

Howard Johnson

Information Intelligence Sciences

Complexity Issues in Security

Howard L. Johnson

Information Intelligence Sciences, Inc.

Presented to:

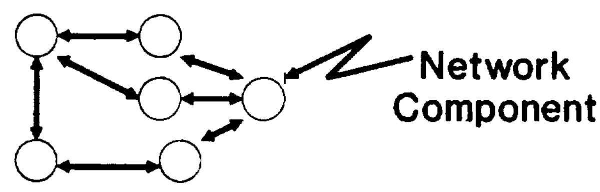
Information Security and Integrity System Symposium

May 16, 1990

Complex System

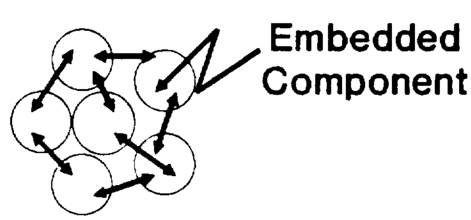
Network

A system composed of connected components



Embedded System

A component that helps comprise a system



Complexity Issues

Policy Complexity

Multiple Security Types
Multiple Security Policies
Interface Policy
Violation of Policy

Architectural Complexity

Phased Build
Distributed Security
Encryption and Unalterable Code
Development

Multiple Security Types

- Sensitivity - Authorized disclosures
- Integrity - Authorized execution of programs and modification of data
- Service Assurance - Authorized and unimpeded proper use of resources
- Safety - Assuring operations without resulting in unacceptable risk

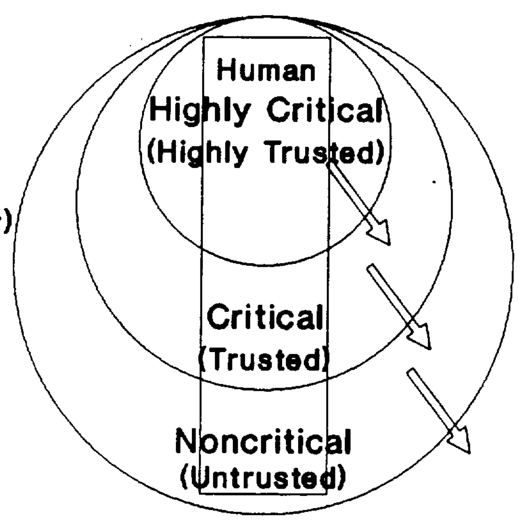
Etc.

Security Based on Mission Criticality

- o Protection for Critical and Highly Critical functions
- o Protection objectives are integrity and service assurance
- o Threat is malicious (e.g., malicious code)
- o Denial-of-service attacks are a subset of integrity attacks if service rules are defined and supported
- o Detection and recovery within a critical time is an acceptable mechanism

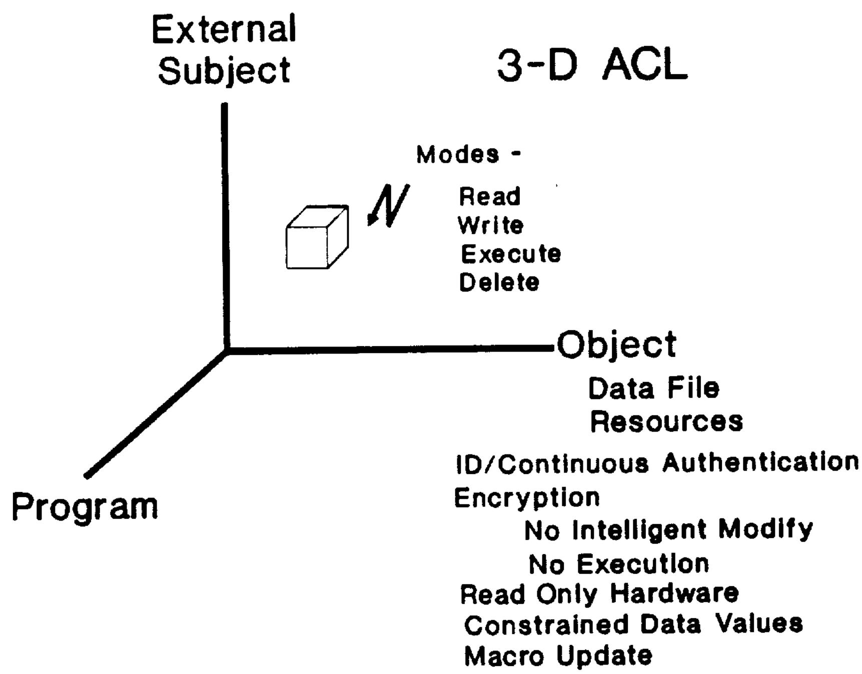
1st Line of Defense - Biba

- o Authorization -
  Command Authority
  via Security Officer
- o Mandatory -
  Trust (Clearance/Other)
- o Discretionary -
  Need to Execute
  Need to Modify
  Capability
  Least Privilege



- o Use Sensitivity Resistance Mechanisms
- o Audit Becomes Detection
- o Each Detection Requires an Action (Recovery)

2nd Line of Defense - Constrained Use



3rd Line of Defense - Detection and Recovery

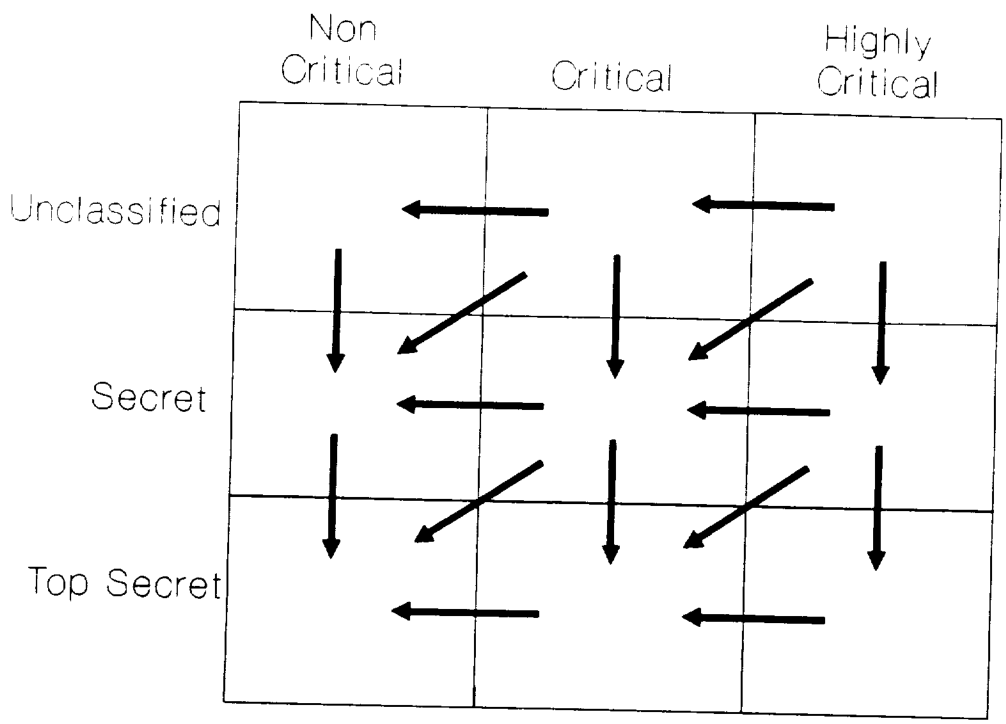
o Detection (Real-Time Audit)

- Intrusion Detection (Deterministic/Inferential)
- Malicious Logic Observables
- Modification Detection (Crypto Checksum)
- Denial of Service Measures
  Monitor Critical Processes
  Target DOS Observables
  Watch Resource Utilization

o Recovery

- Guided by Critical Time
- Remove Threat, Re-Execute
- Alternative/Reconfigure
- Cold/Hot Backup
- Checkpoint Restart
- Manual Intervention
- Distributed Control
- Redundancy/Fault Tolerance

Combined Policy Lattice

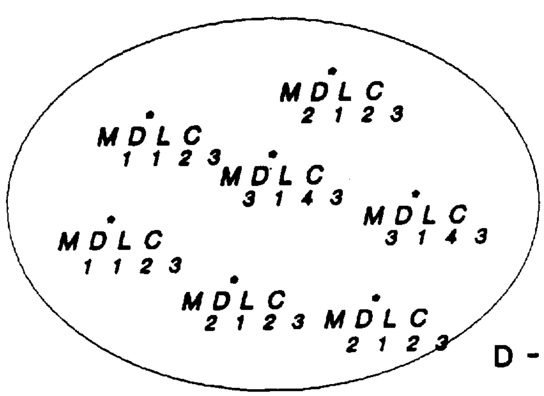


Criteria

	Sensitivity (Orange Book)	Criticality (Integrity and Denial of Service)
Policy		
DAC		Constrained Use
Object Reuse		X
Labels		X
MAC	Bell-La Padula	Biba
Accountability		
ID/Auth		X
Audit		Detection/Availabilit
Availability Assurance		
Operational		
Sys Arch		X
Sys Int		X
Covert Channel	Bell-La Padula	Biba
Trusted Fac Mgmt		X
Trusted Recov		Recovery within critical time
Life Cycle	TCB	All
Test		X
Design Spec & Ver		X
Config Mgmt		X
Trusted Distr		X
Documentation		X

Multiple Security Policies

Point Security Policy



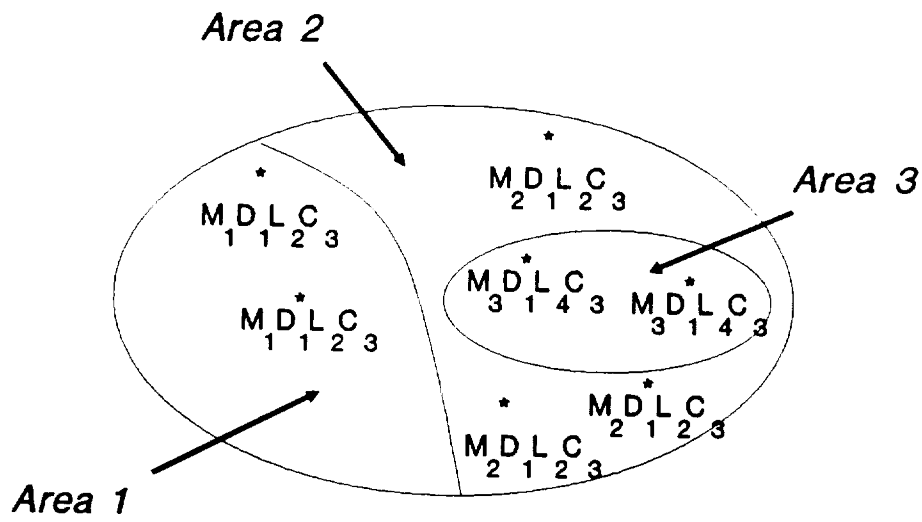
M - Security Mode

D - Orange Book Division/Class

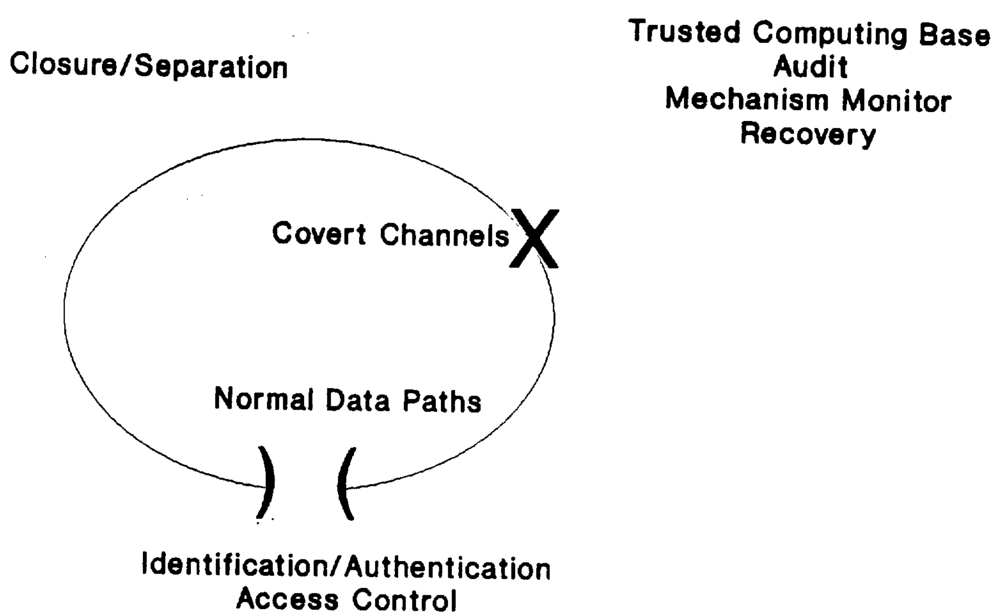
L - Highest Classification Level at that Point

C - Lowest User Clearance Level Allowed Access

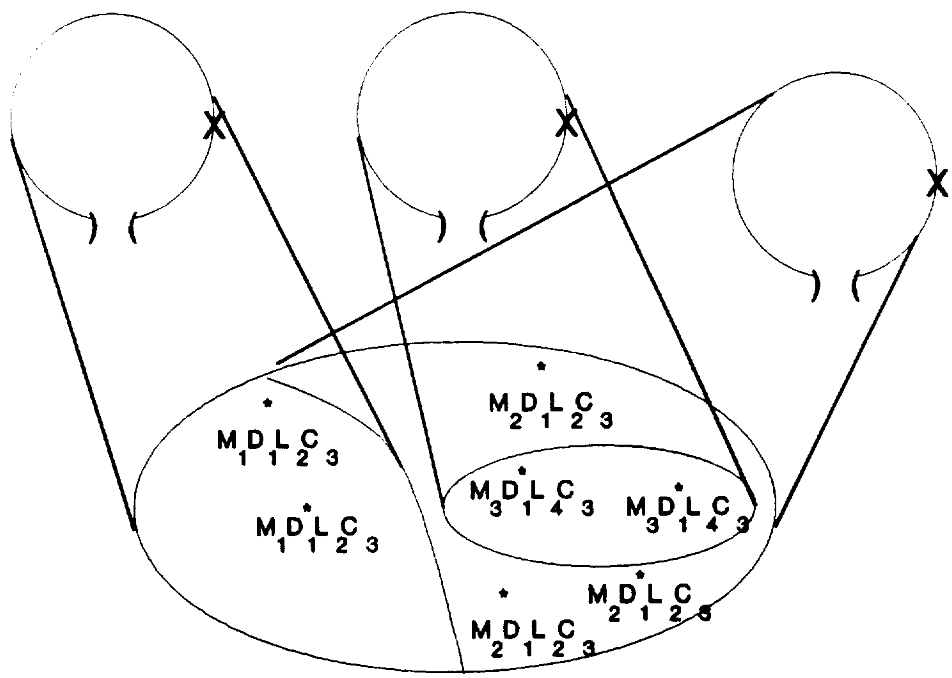
Constant Policy Domains



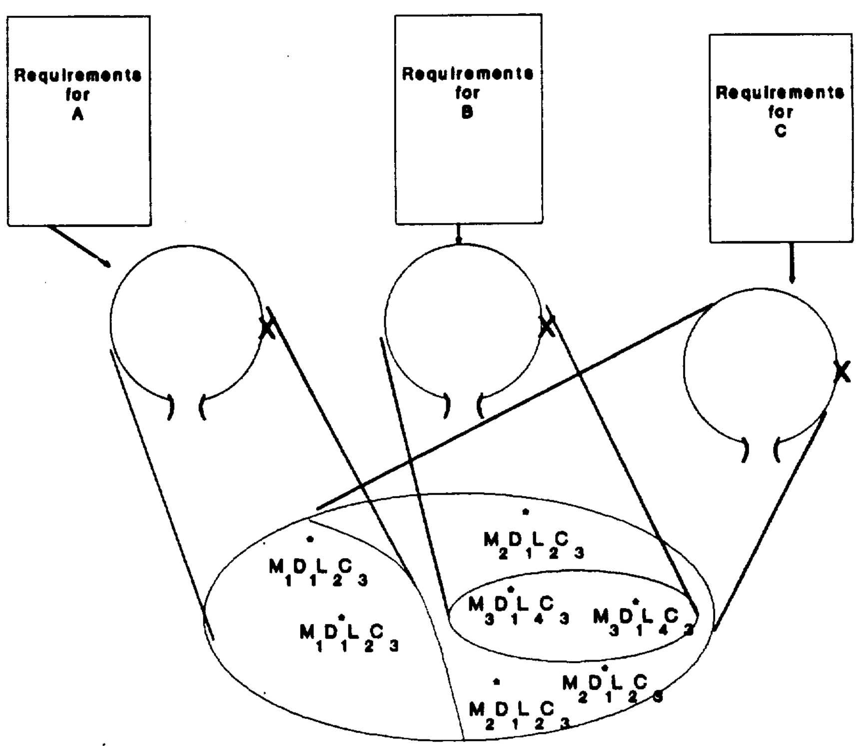
Mechanisms in Constant Policy Domain



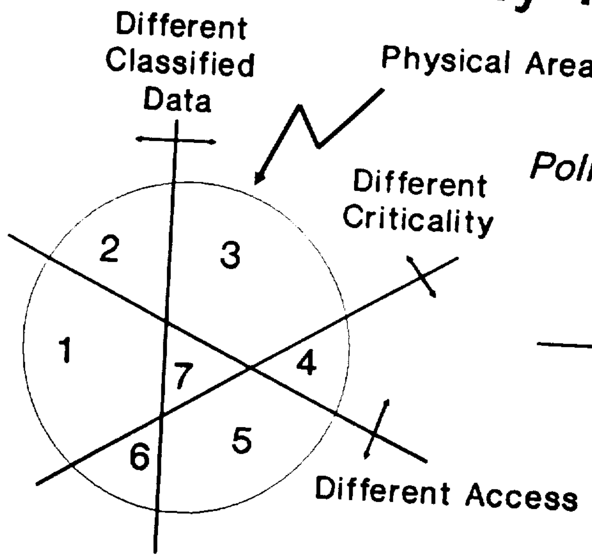
Mechanisms and Constant Policy Domains



Summary of Relationships

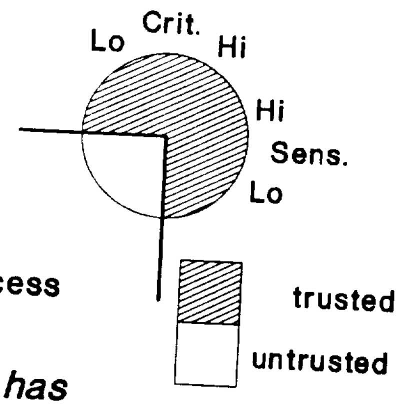


Multiple Security Policies



7 different policies - because each has a different risk

Policy Simplification



2 policies

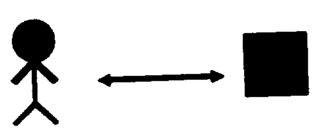
$ overkill

Interface

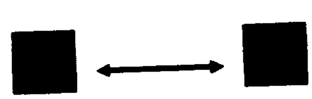
Interface Policy

o Communication between two nodes
  Separate control
  Different constant policy domains

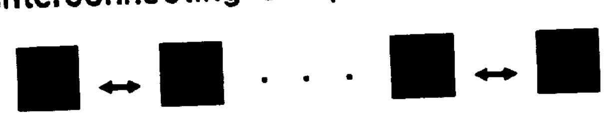
Person/Component



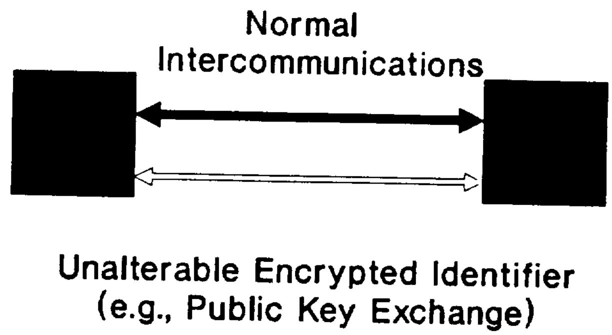
Component/Interconnecting Component



Through Interconnecting Components



Component Authentication



Interface Control

Trust - Limit exchange

Transform Levels

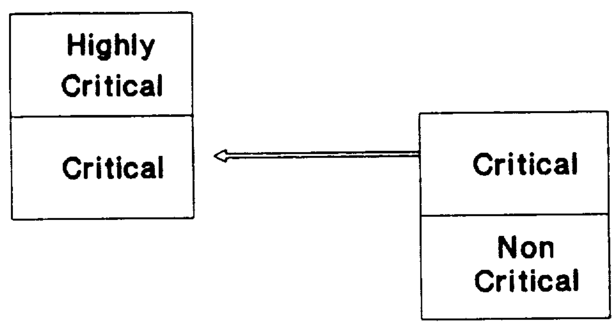
Validate data

Exposure - Reflect inherent exposure

Propagate cascading exposure

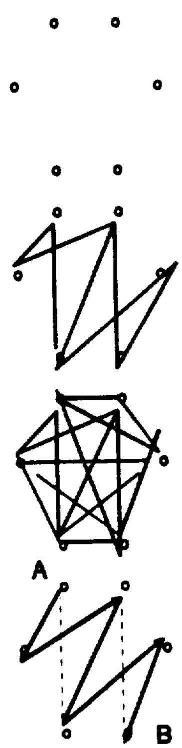
Cascading Problem

Connectivity Increases Exposure and Therefore Risk



Combinatorics

Nodes = N



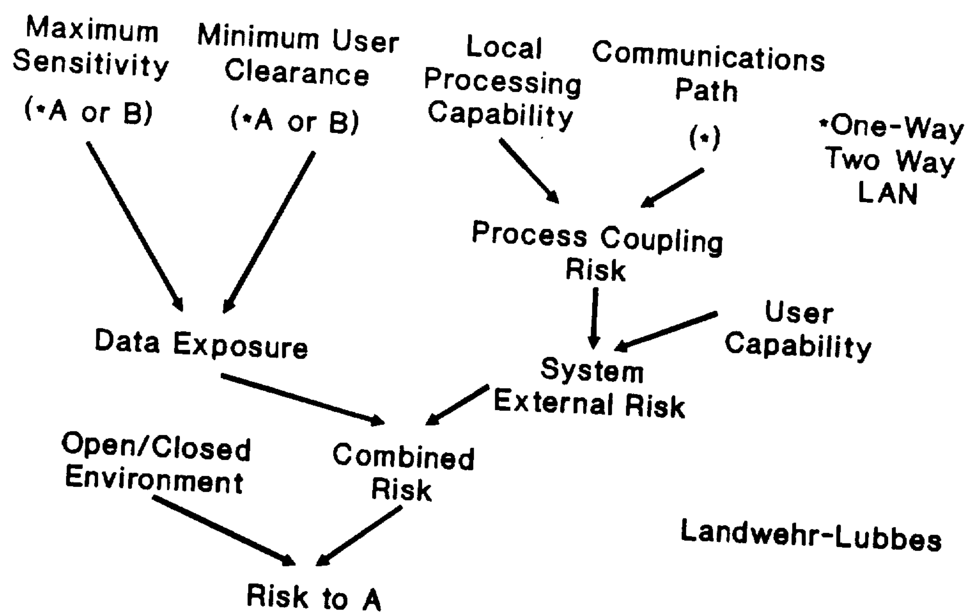
Connectivity $O(N)$ to $O(N^2)$

Security Policy Interface Encrypted $O(N^2)$

Possible Paths (Security Policy Interface Unencrypted) $O(1)$ to $O(N^3)$

Complex System Evaluation

Every node pair wrt every path (A wrt B)



Security Beyond Alphanumerics

Directed to human user
- Image
- Computer generated voice

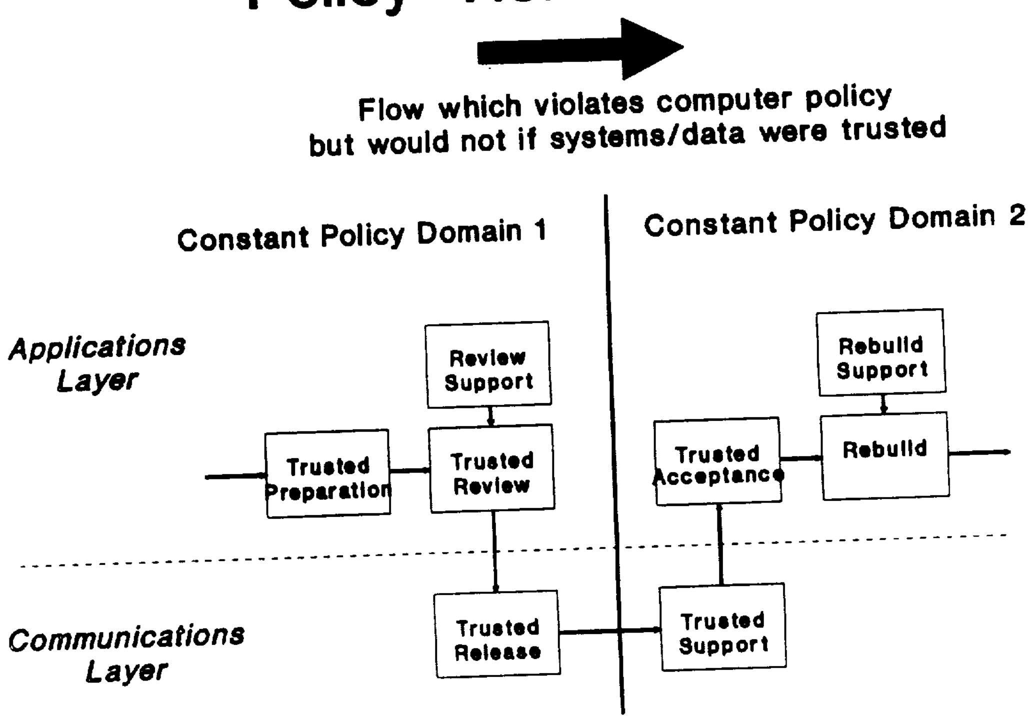
Not necessarily directed to human
- Action (e.g., electromechanical)
- Function
- Characteristic (e.g., traffic)

Human user interaction
- Non electronic authentication
- Communicate classification
- Communicate classification change

Autonomous
- Classification masking
- Deception strategies

Policy Violation

Policy Violations



Policy Violation Functions

Sanitization

Trusted Preparation

- Reduce data, maintain information
  (Share apriori data, send updates)
- Put in Optimum Form For
  - Manual Review
  - Software Review
- Encased Cryptographic Checksum

Review Support

- Knowledge Base
  - What info should be
  - What info shouldn't be
- Placed in
  - Manual Form
  - Software Form

Trusted Review

- Qualified Reviewers
- Ensure all and only all data available
- Review support data separated from data

Trusted Release

- Play back

Trusted Receipt

- Covert Channel

Trusted Acceptance

- Authenticate validity & currency

Reconstruction

- Expand data to usable form

Reconstruction Support

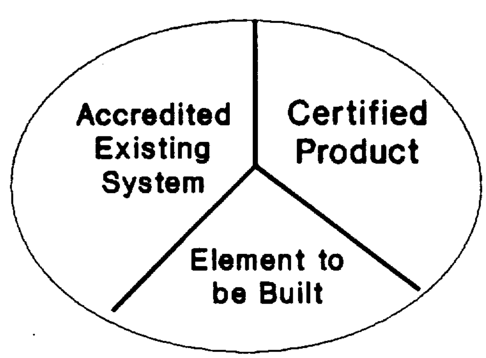
- Apriori info
- Baseline for update

PHASED BUILD

Phased Build

Within a Domain of Constant Policy

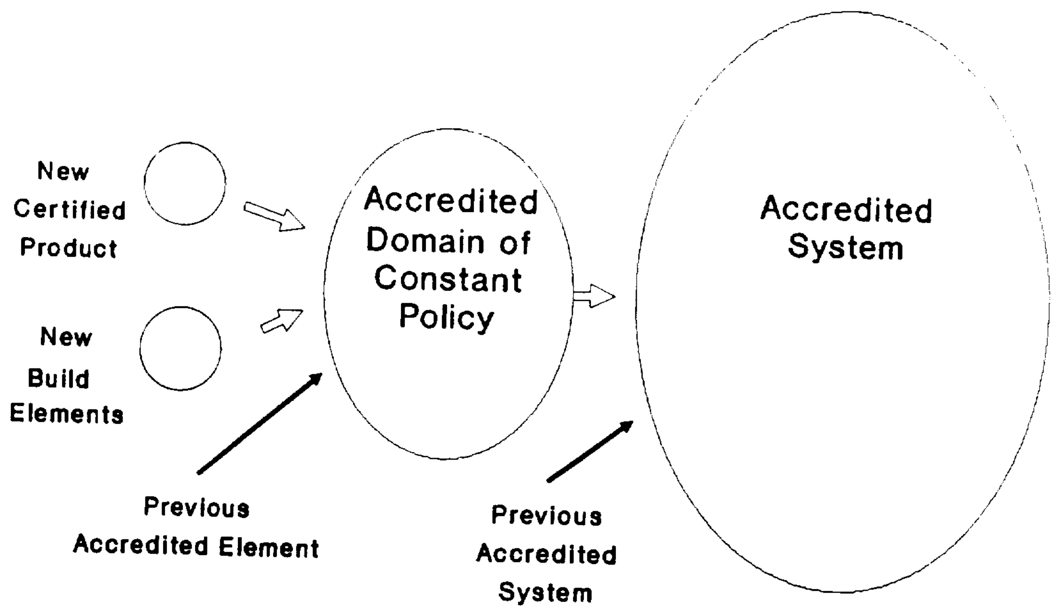
No matter how big, if it still meets goals



Certified by NCSC or equivalent process

Decomposed to the Point where all security relevant properties are visable

Progressive Build



Encryption and Unalterable Coding

Encryption and Coding Uses

Non Disclosure (NSA controlled)

Other (NSA advisory or control)

- Identification (components)
- Authentication (components)
- Key Management
- Labeling
- Mechanism Protection
- Modification Detection
- Trust Retention
- Bandwidth Filling (Covert Channel)
- Execution Prevention
- Intelligent Change Prevent
- Enemy Spoofing
- Signature
- Notarization

Distributed Security

Distributed Security

System 1

Ref Monitor
R/T Audit
Recovery

System 2

Ref Monitor
R/T Audit
Recovery

.
.
.

System n

Ref Monitor
R/T Audit
Recovery

Secure
LAN

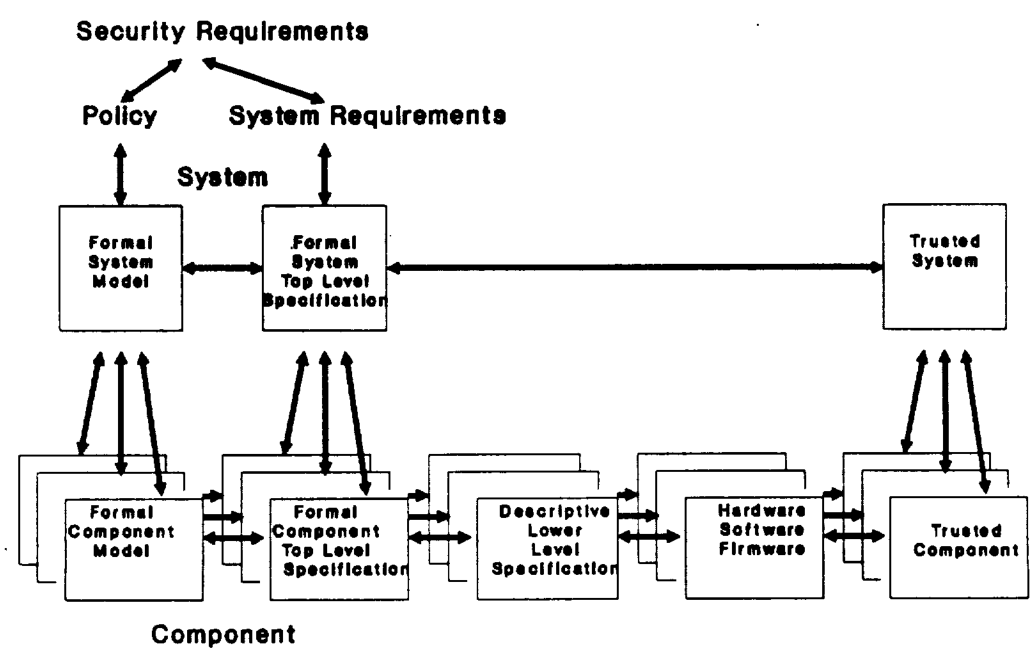
Distributed
Database
and
Synchronization
Techniques

Security Elements

- o Identification
- o Authentication
- o Access Control Data
- o Key Distribution
- o Historical Audit
- o Upgrade/Downgrade Guard
- o Multi System Alarm
- o Configuration Control
- o Security Officer

Development

Development



Order of Development

Object	Develop	Simplify	Iterate
Policy	*	*	
Elementary Components	*	*	*
Constant Policy Domains	*	*	*
Interface Policy	*	*	*
Security Model	*	*	*
Security Architecture	*	*	*
System Architecture	*	*	*
Development	*		

Security in Computer Networks

Colin Rous

Digital Equipment Corporation

Notes

Notes

[redacted] Lunch Speaker

Ethics: Mandate VS. Choice

Marlene Campbell

Murray State University

268-269-270

Notes

Notes

Common Session

Computer Viruses

Angel Riveria
Sector Technologies, Inc.

272-273-274

PRECEDING PAGE BLANK NOT FILMED

Notes

Notes

Panel discussion on NASA concerns related to trusted computing support for life and property critical systems

276-277-278

PRECEDING PAGE BLANK NOT FILMED

Notes

Notes

Notes

Notes